Lappeenrannan teknillinen yliopisto
School of Business and Management
Tietotekniikan koulutusohjelma

Kandidaatintyö

Samuli Siitonen

Hakuperustainen proseduraalinen pelien sisällön generointi geneettisellä algoritmilla

Työn tarkastaja: Tutkijaopettaja (TkT) Jouni Ikonen

Työn ohjaajat: Tutkijaopettaja (TkT) Jouni Ikonen
TkT Jussi Kasurinen

29.6.2017

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
School of Business and Management
Tietotekniikan koulutusohjelma

Samuli Siitonen

Hakuperustainen proseduraalinen pelien sisällön generointi geneettisellä algoritmilla

Kandidaatintyö

2017

53 sivua, 8 kuvaa, 10 taulukkoa

Työn tarkastaja: Tutkijaopettaja (TkT) Jouni Ikonen

Hakusanat: proseduraalinen sisällön generointi, hakuperustainen proseduraalinen sisällön generointi, geneettinen algoritmi, evolutionäärinen algoritmi, pelisisältö, pelit
Keywords: procedural content generation, search-based procedural content generation, genetic algorithm, evolutionary algorithm, game content, games

Tässä työssä selvitetään, miten geneettisillä algoritmeilla voidaan generoida kaksiulotteisia sokkelomaisia pelikenttiä proseduraalisen sisällön generoinnin näkökulmasta. Selvitykselle luotiin perusta sisällön generointiin ja geneettisiin algoritmeihin liittyvän teoriatiedon läpikäymisen avulla. Manuaalisen sisällön tuottamisen kustannukset, uusien pelityyppien syntyminen ja mielikuvituksen augmentointi toimivat merkittävinä proseduraalisen sisällön generoinnin hyödyntämisen motivaattoreina. Työn tuloksena saatiin luotua 2-ulotteisia sokkeloita generoiva geneettinen algoritmi, jonka toteuttaminen yhdistyi työssä läpikäytyihin proseduraalisen sisällön generoinnin tutkimuksiin.

ABSTRACT

Lappeenranta University of Technology
School of Business and Management
Degree Program in Computer Science

Samuli Siitonen

Search-Based Procedural Game Content Generation with a Genetic Algorithm

Bachelor's Thesis

53 pages, 8 pictures, 10 tables

Examiner : Associate Professor (D.Sc.) Jouni Ikonen

Keywords: procedural content generation, search-based procedural content generation, genetic algorithm, evolutionary algorithm, game content, games

This thesis contains an examination about how genetic algorithms can be used to generate two-dimensional maze-like game maps from the viewpoint of procedural content generation. The groundwork for this examination was created by going through theoretical knowledge related to procedural content generation and genetic algorithms. Expenses of manual content creation, birth of new game types and augmenting imagination function as remarkable motivators for utilizing procedural content generation. As a result of this thesis, a genetic algorithm for generation two-dimensional mazes was able to be created. Implementing this algorithm connects to a previous part in this thesis that consisted of going through research papers on procedural content generation.

ALKUSANAT

Haluan kiittää Jouni Ikosta työni loppuun ohjaamisesta ja Jussi Kasurista, joka toimi työni ohjaajana sen työstämisen alkuvaiheiden aikana. Kiitän myös kaikkia minua tämän työn tekemisen aikana tukeneita henkilöitä.

SISÄLLYSLUETTELO

SYMBOLI- JA LYHENNELUETTELO

APS	Answer Set Programming
ASM	Attribute Similarity Measure
CA	Cellular Automaton
EA	Evolutionary Algorithm
EC	Evolutionary Computing
EP	Evolutionary Programming
ES	Evolution Strategies
FPS	Fitness Proportional Selection
GA	Genetic Algorithm
GP	Genetic Programming
NPC	Non-Player Character
PCG	Procedural Content Generation
PRNG	Pseudo-Random Number Generator
SBPCG	Search-Based Procedural Content Generation
SGA	Simple Genetic Algorithm

1 JOHDANTO

Tämä kandidaatintyö käsittelee proseduraalista sisällön generointia (engl. procedural content generation, PCG) tietokonepelien näkökulmasta. Käsittelyssä painotetaan hakuperustaista PCG:tä (engl. search-based procedural content generation, SBPCG), sen tieteellistä tutkimusta ja toteuttamista geneettisillä algoritmeilla (engl. genetic algorithm, GA).

1.1 Tausta

Tämän työn tarkoituksena on yhdistää PCG:n ja erityisesti SBPCG:n teoria käytäntöön. Työssä käydään läpi PCG:n teoreettinen perusta, siirrytään GA:iden teorian esittämiseen, esitellään SBPCG ja toteutetaan yksinkertainen GA SBPCG:n ilmentymänä.

Sisällön tavanomaisten tuottamismekanismien estävät (engl. prohibitive) kustannukset (Togelius et al. 2011b:172) toimivat yhtenä tämän työn ja PCG:n hyödyntämisen motivaattorina. Tämä perustuu kustannusten ilmeiseen merkitykseen uuden peliliiketoiminnan perustamisen rajoittimina. Muita motivaattoreita ovat PCG:n potentiaalisesti mahdollistama uusien ja uudelleenpelattavuudeltaan loputtomien pelien luominen sekä ihmisen luovuuden tehostaminen (Togelius et al. 2011b:172-173). Näillä tekijöillä voidaan nähdä positiivinen vaikutus pelaajien ja pelikehittäjien toimintaan parempien pelikokemusten ja tehostetun työskentelyn valossa.

1.2 Tavoitteet ja rajaukset

Tällä työllä on seuraava tutkimuskysymysten sarja: Miten geneettisillä algoritmeilla voidaan vastata valittuun sisällön generoinnin ongelmaan (2-ulotteiset sokkelomaiset pelikentät) proseduraalisen sisällön generoinnin näkökulmasta? Minkälainen tämän ongelman ratkaiseva geneettinen algoritmi on komponenttiensa ja generoidun sisällön luonteen osalta? Mitä geneettisen algoritmien parametrien rajattu vaihtaminen saa aikaan generoidun sisällön kannalta?

Esitettyihin tutkimuskysymyksiin vastaaminen perustuu GA:iden, PCG:n, GA:iden ja PCG:n suhteen käsittelemiseen sekä GA:n toteutustyön esittämiseen. Toteutustyötä

edeltävät käsittelyvaiheet toimivat toteutustyön pohjana ja ne mahdollistavat sen suoraviivaisen läpikäymisen valmiin teoriatiedon tarjoamisen kautta.

Tämän työn on tarkoitus toimia yleisluontoisena PCG:n esittelytekstinä, joka tarkentuu yhden sen alaluokan (SBPCG) käsittelemiseen ja toteuttamiseen geneettisenä algoritmina Go-ohjelmointikielellä. Työ tarjoaa syventymismahdollisuuden PCG:hen SBPCG:n tarkastelun ja esimerkinomaisen toteutustyön toteuttamisen kautta.

Käsittelyssä tullaan välttämään PCG:n ja SBPCG:n ymmärtämistä vaikeuttavien teoriatietojen esittäminen. Tämä koskee erityisesti geneettisten algoritmien teoriaa ja niiden toteuttamiseen liittyviä huomioita; geneettinen algoritmi tulee vastaamaan suurilta osin yksinkertaista geneettistä algoritmia (engl. simple genetic algorithm, SGA). Luotavan GA eroaa SGA:asta merkittävimmin erilaisen valintamekanismin ja samanaikaisen suorittamisen hyödyntämisen osalta. PCG:n osalta esitetään yleinen, myöhempää SBPCG:n tarkastelua tukeva teoria, ja SBPCG:n teoriakatsaus puolestaan tukee yksinkertaisen toteutustyön esittämistä. GA:ista esitetään toteutustyön kannalta oleelliset perusominaisuudet; tämä kattaa muun muassa GA:n peruskomponentit, jotka yhtyvät osittain yksinkertaisen GA:n kuvaukseen.

1.3 Työn rakenne

Tämä työ jakautuu pääluvuittain PCG:n yleisteoriaan (luku 2), GA:iden perusteiden läpikäyntiin (luku 3), SBPCG:n yleisteoriaan ja tutkimukseen (luvut 4 ja 5) ja GA:n toteuttamistyöhön (luku 6). Lopuksi esitetään pohdinnan ja tulevaisuuden käsittely (luku7).

Toinen luku käsittelee PCG:tä sen määrittelemisen, taustan, tutkimuksen, tarjoaminen mahdollisuuksien, taksonomian ja generointimenetelmien osalta. Kolmas luku kattaa lyhyen esittelyn GA:iden taustaan, jonka jälkeen käsitellään GA:iden roolia evoluution abstraktioina. Neljäs luku aloittaa PCG:n ja GA:iden yhdistämisen GA:iden PCG:ssä käytettävien kelpoisuusfunktiotyyppien läpikäymisellä. Viiden luku käsittelee GA:ita SBPCG:n tutkimuksissa; esitellään kaksi SBPCG:n GA:n toteutusta 2-ulotteisten sokkelomaisten pelikenttien generoimiseen ja vertaillaan kyseisiä tutkimuksia. Kuudes luku sisältää GA:n toteuttamisen Go-ohjelmointikielellä; GA:lla generoidaan viidennen

luvun tapaan 2-ulotteisia sokkelomaisia pelikenttiä. Toteutustyön osalta esitetään algoritmin komponenttien valitseminen, toteutustyön vaiheet ja sisällön generoinnin tulokset. Seitsemäs luku kattaa työn pohdinnan ja katselmuksen tulevaisuuteen.

2 PROSEDURAALINEN SISÄLLÖN GENEROINTI PELEISSÄ

Proseduraalista sisällön generointia (PCG) käsitellään seuraavissa alaluvuissa sen pääpiirteitten osalta. Käsittelyssä tullaan painottamaan PCG:n tutkimusperustaista näkökulmaa. Tämän pääluvun tarkoituksena on tarjota PCG:n teorian kokonaiskuva, jolla tuetaan työn muiden pääaiheiden käsittelyä.

2.1 Pelien proseduraalisen sisällön generoinnin määritteleminen

Noor et al. (2016) mukaan proseduraalisuuden käsite merkitsee tietokoneproseduureja tai algoritmeja. Proseduraalinen generointi voidaan puolestaan mieltää proseduurien tai algoritmien hyväksikäyttämiseksi jonkin tuloksen luomiseksi. Noor et al. (2016:2)

Togelius et al. (2011a) mukaan PCG voidaan tietyin varauksin määritellä algoritmiperustaiseksi pelin sisällön luomiseksi. Tässä käyttäjällä on rajoitettu tai epäsuora vaikutus generointiin algoritmin syötteiden lähteenä. Heidän mukaansa PCG:n konseptilla on kuitenkin sumeat ja epäselvät rajat. Lisäksi he ovat kertoneet kaikkien hyväksymän määritelmän tuottamisen toivottomuudesta, mikä johtuu PCG:n käyttäjäryhmien (muun muassa tutkijat ja pelisuunnittelijat) perspektiivieroista.

Hendrikx et al. (2013:1:4-1:8) mukaan pelien sisältö voidaan jakaa kuuteen luokkaan ja he ovat esittäneet näihin luokkiin perustuvan virtuaalisen pyramidin. Tässä pyramidissa alemmat sisältötasot voivat toimia ylempien osana; esimerkiksi peli bittejä (muun muassa tektstuurit, äänet ja kasvillisuus) voidaan käyttää peliavaruuden tasossa (heti pelibittien yläpuolella oleva taso) ulkotilan karttojen (engl. outdoor maps) muodostamiseen.

Taulukossa 1 on esitetty Hendrixk et al määrittelemät sisältöluokat heidän luomansa pyramidin tasojen mukaisessa järjestyksessä. Tässä taulukossa ylimpänä rivinä toimii johdettu sisältö, joka voi koostua esimerkiksi pelaajan toimintaa käsittelevistä pelinsisäisistä uutisista. (Hendrikx et al. 2013).

Taulukko 1. Pelien sisältöluokat Hendrixk et al. (2013:1:4-1:8) perusteella.

Sisältöluokka	Sisältöesimerkki
johdettu sisältö (engl. derived content)	pistetilasto
pelisuunnittelu (engl. game design)	pelimaailman suunnitteleminen
peliskenaariot (engl. game scenarios)	tarinat ja pelitasot
pelisysteemit (engl. game systems)	ekosysteemit ja entiteettien käyttäytyminen
peliavaruus (engl. game space)	karttat: ulko- ja sisätilat
pelibitit (engl. game bits)	tekstuurit ja kasvillisuus

Togelius et al. (2011b) ovat määritelleet pelisisällöksi kaikki pelin näkökohdat, jotka vaikuttavat pelattavuuteen (engl. gameplay), pois lukien tietokoneen ohjaamien hahmojen (engl. non-player character, NPC) käyttäytyminen ja pelimoottori. Mahlmann, Togelius ja Yannakakis (2012) ovat lisäksi maininneet, että käyttäytymisen toimintatapojen generoiminen voidaan laskea PCG:ksi joissakin tapauksissa. Togelius et al. (2011b) esittämä sisällön määritelmä kattaa muun muassa pelin kartat ja tarinat, ja samat termit esiintyvät myös Hendrikx et al. (2013) luomassa virtuaalisessa pyramidissa.

2.2 Pelien proseduraalisen sisällön generoinnin tausta ja tutkimuksen nykytilanne

PCG esiintyi ensimmäistä kertaa pelikäytössä 1980-luvulla, jolloin sitä käytettiin Rogue-pelissä automaattiseen tasojen generointiin (Yannakakis & Togelius 2011:148). Muita sen peliesimerkkejä ovat muun muassa Elite (1984) (Yannakakis & Togelius 2011:148), Civilization (1991), Diablo (1996), Borderlands (2009) (Togelius et al. 2011a:1) ja No Man’s Sky (2016). PCG:tä käytetään nykypäivänä myös systeemitasolla; esimerkiksi SpeedTree-systeemi (PCG-systeemi) mahdollistaa pelien kasvillisuuden generoinnin (Togelius et al. 2013:61-62).

Togelius et al. (2013:62) esittämien huomioiden perusteella PCG:n tiedeyhteisöä voidaan kuvata varsin nuoreksi. Heidän mukaansa ensimmäinen PCG:lle omistettu työpaja järjestettiin vuonna 2009 ja sille kokonaan omistettu erikoisjulkaisu sijoittui vuodelle 2011.

PCG:n tutkimuskentän nykytilasta (2016) on tehty katsaus Procedural Content Generation in Games –kirjassa (Shaker et al. 2016). Kyseisessä teoksen mukaan PCG:n tutkimustyön määrä on kasvanut runsaasti viime vuosina (Shaker et al. 2016:5). Yhtenä tämän väitteen todentajana toimii PCG:n tavoitteiden, haasteiden ja mahdollisuuksien tieteellinen käsitteleminen (Togelius et al. 2013). Mainitut tavoitteet on esitelty taulukossa 2.

Taulukko 2. PCG:n tutkimuksen kolme tavoitetta Togeliuksen et al (2013:62-64) perusteella.

Tavoite	Selitys
monitasoinen ja -sisältöinen PCG (engl. multi-level multi-content PCG)	Kokonaisen pelimaailman generointi jossa pelin pelimoottori voi toimia vaihdettavana komponenttina (pelimoottoria ei generoida) Pelimoottori tarjoaa pelin primitiivejä (muun muassa liikkuminen) ja sen pystyvyys (engl. capabilities) on ainoana ilmaistavien mahdollisuuksien (engl. expressive space) rajoitteena.
PCG-perustainen pelisuunnittelu (engl. PCG-based game design)	Sisältää pyrkimyksen PCG:n ja pelien välisen suhteen tiivistämiseen. Tässä pelin pelityyppi perustuu PCG:hen, joka myös toimii pelin keskeisenä mekaniikkana; pelin sisältö on loppumatonta ja sen tutkiminen toimii pelikokemuksen ytimenä.
kokonaisten pelien generointi (engl. generating complete games)	Sisältää miellyttävän ja samalla aikaa täysin uuden pelin generoimisen. Tässä generoidaan myös pelin pelimoottori, joka toimii yhtenä erottavana tekijänä monitasoisen ja –sisältöisen PCG:n päämäärään.

Taulukon 2 PCG.n tutkimuksen tavoitteet ovat osa suurempaa Togeliuksen et al. (2013) esittämien pohdintojen sarjaa. He ovat esittäneet yhdeksän näiden tavoitteiden saavuttamiseksi kohdattavaa haastetta (Togelius et al. 2013:64-70) ja viisi kehitysaskelta näiden haasteiden voittamiseksi (Togelius et al. 2013:70-73).

Teoreettisen pohdinnan lisäksi varsinaista pelien sisällön generointia on tutkittu ansiokkaasti; tämän esimerkkejä ovat muun muassa loputtomien luolastojen (Johnson et al. 2010) ja tietokoneen ohjaamien pelaajien käyttäytymisen (Headleand et al. 2015) generoiminen.

2.3 Pelien proseduraalisen sisällön generoinnin tarjoamat mahdollisuudet

Togelius et al. (2011b) ovat listanneet neljä pelinkehittäjien kannalta olennaista syytä PCG:hen tutustumiseen:

- Muistin säästäminen.
- Manuaalisen sisällön tuottamisen suuret kulut.
- Uusien PCG:hen perustuvien pelityyppien esiin nouseminen.
- Ihmisen mielikuvituksen augmentointi.

Ensimmäiseksi listattu muistin säästäminen ei alkuun vaikuta tärkeältä nykypäivänä vallitsevan muistin runsauden valossa. Togeliuksen et al. (2011b) mainitsema Elite-pelin esimerkkitapaus kuitenkin tarjoaa tähän motivaattorin; kyseisessä pelissä satoja tähtijärjestelmiä säilytettiin kymmenien kilotavujen suuruusluokassa. Tarpeeksi laadukkaan pelikokemuksen tarjoaminen generointialgoritmin siemenluvun muodossa saattaisi lyhentää pelien latausaikoja ja täten parantaa digitaalisten pelikauppojen asiakaskokemuksia.

Toiseksi listattu manuaalisen sisällön tuottamisen suuret kulut voi toimia merkittävä estotekijänä juuri toimintaansa aloittavien peliyritysten kilpailukyvyn suhteen. Tämän voidaan nähdä yhdistyvän kolmanteen ja neljänteen PCG:n käytön motivaattoreihin; pelinkehittäjät voisivat keskittyä niihin pelinkehityksen osa-alueisiin, joita ei olla tehostettu. Tämä voidaan yhdistää ohjelmointikielten aikajanaan, jossa yhä korkeammat abstraktiotasot tarjoavat kompleksisuuden rajoittamismahdollisuuksia. Kohdattavan kompleksisuuden kohtaamisen välttäminen voisi näkyä tulosten kasvuna etenkin generointialgoritmeille tarjotun laskentatehon kasvaessa.

Edellä esitetty pohdinta manuaalisen sisällön tuottamisen suurista kustannuksista

pienyritysten kannalta voimistuvat Togeliuksen et al. (2011b) esittäminen pohdintojen voimin. Heidän mukaansa laitteiston kehitysaskelien myötä pelien yksityiskohtien tason ja määrän vaatimukset tulevat kasvamaan. Tämä on huomattavissa etenkin suurelle pelaajayleisölle suunnattujen pelijulkaisujen mittakaavassa ja graafisen esityksen laadussa, johon pienet peliyritykset eivät mitä luultavammin pysty vastaamaan.

Togelius et al. (2011b) ovat myös esittäneet pohdintoja uusien pelityyppien syntymiseen liittyen; he mainitsevat muun muassa pelikokemuksiltaan loppumattomien pelien syntymisen. Tämä voisi toimia hidastimena peleihin kyllästymiselle ja tästä seuraavalle yleisön haluna kalliiden sekä riskialttiiden pelien kehittämiselle. Sisällöltään loppumattomien pelien suhteen pelikehittäjien täytyisi todennäköisesti löytää uusia ansaintamalleja, jotka voisivat korvata tarpeen jatkuvalle uusien pelien nopealle työstämiselle.

Edellä esitetty potentiaalinen tapahtumaketju voisi jatkua pisteeseen, jossa yhä useammilla ihmisillä olisi mahdollisuus pelattavuudeltaan korkealaatuisten tuotosten saavuttamiseen. Tällöin pelien kehittämisestä voisi muodostua helposti lähestyttävä toimiala, joka pystyisi ottamaan käyttöönsä runsaan määrän luovaa ja luovuudeltaan augmentoituja kehittäjiä laajasta ihmiskirjosta. Tämä yhdistyy Togeliuksen et al. (2011b) pohdintaan, jossa he toteavat pelisuunnittelijoiden kykyjen vaihtelun ja niiden augmentoinnista saavutettavan hyödyn.

2.4 Pelien proseduraalisen sisällön generoinnin taksonomia

Togelius et al (2016) esittäneet PCG-menetelmille taksonomian, joka on uusittu version Togeliuksen et al. (2011b) aikaisemmin laatimasta taksonomiasta. Tässä uusitussa taksonomiassa generointimenetelmät on jaettu taulukon 2 mukaisesti seitsemään ulottuvuuteen. Näissä ulottuvuuksissa generointimenetelmät sijoittuvat ulottuvuuksien ääripäinen väliin.

Taulukko 3. Uusitun PCG:n taksonomian seitsemän ulottuvuutta ääripääesityksenä Shaker et al. (2016:7-10) perusteella.

Ulottuvuus	1. ääripää	2. ääripää	Selitys
1.	online	offline	Sisällön generointi pelaamisen (online) tai pelinkehityksen (offline) aikana.
2.	välttämätön (engl. necessary)	valinnainen (engl. optional)	Generoidaan pelattavuuden kannalta välttämätöntä tai avustavaa sisältöä.
3.	satunnainen siemenluku (engl. random seed)	parametrit (engl. parameters)	Sisällön generoinnin kontrolloimisen aste ja ulottuvuudet; siemenluvulla vähäinen ja parametreilla oletettavasti moniulotteinen kontrolloitavuus.
4.	geneerinen (engl. generic)	mukautuva (engl. adaptive)	Sisältö generoidaan pelaajan toimista riippumattomasti (geneerisesti) tai se perustuu pelaajan valintoihin.
5.	stokastinen (engl. stochastic)	deterministinen (engl. deterministic)	Sisältöä voidaan generoida satunnaisesti ja täten toistamattomasti tai deterministisesti, jolloin samoilla syötteillä generoidaan samaa sisältöä.
6.	rakentava (engl. constructive)	generoi ja testaa (engl generate and test)	Sisältö voidaan generoida yhdellä kertaa (rakentavasti) tai se voi olla lopputulos useiden generointi- ja testauskertojen prosessille.

7.	automaattinen generointi (engl. automatic generation)	sekoitettu luominen (engl. mixed authorship)	Sisältöä generoidaan syötteiden antamisen jälkeen ilman pelaajan tai suunnittelijan vaikutusta (automaattinen generointi), tai sisältö voi perustua ihmisen ja algoritmin yhteistyöhön (sekoitettu luominen).

Taulukosta 2 voidaan erottaa kaksi ulottuvuusjoukkoa:

- Joukko 1 {1, 4, 7}
- Joukko 2 {2, 3, 5, 6}

Joukko 1 koostuu ulottuvuuksista, jotka voidaan mieltää olennaisiksi kun tarkastellaan pelaajan ja PCG-menetelmän välistä suhdetta. Tässä ensimmäisen ulottuvuuden online-ääripää voidaan nähdä yläluokkana neljännen ulottuvuuden mukautuvalle generoinnille ja ulottuvuuden 7 sekoitetulle luomiselle. Näihin ulottuvuuksiin ja mainittuihin ääripäihin sopiva PCG-menetelmä toimisi pelin sisällön muovaajana, joka mukautuisi ja tekisi yhteistyötä pelaajan kanssa. Näin pelaaja toimisi oman pelikokemuksensa arkkitehtina, vaikka generointialgoritmi ja sen toiminta olisi hänen näkökulmastaan musta laatikko.

Joukko 2 koostuu ulottuvuuksista, joiden käsittelemisellä voidaan saada selvyyttä PCG-menetelmän generoivan sisällön laadun takaamiseen. Tässä toisen ulottuvuuden välttämättömän sisällön ääripää yhdistyisi kolmannen tason parametrien ääripäähän sisällön laadun takaamiseksi. Tämä yhteys perustuisi pelikehittäjien tarpeeseen tarjota toimiva pelikokemus, jossa pelissä eteneminen varmistettaisiin. Viidennen ulottuvuuden stokastisuuden ja kuuden ulottuvuuden generoi ja testaa -ääripäät toimisivat vuorostaan yhteistyössä uudenlaisen ja pelattavuudeltaan toimivan sisällön löytämisessä. Mainittujen joukon 2 alajoukkojen mukainen PCG-menetelmä pyrkisi toimivan, laadukkaan ja uudentyyppisen sisällön generoimiseen.

Hendrixk et al (2013:1:8) esittämä PCG-G:n (engl. procedural content generation for

games) taksonomia on esitetty taulukossa 4. Heidän luomassaan taksonomiassa esiintyy viisi generointimenetelmien luokkaa, mikä poikkeaa Shakerin et al (2016:7-10) uloteperustaisesta taksonomiasta. Hendrixk et al ottavat suoraan kantaa menetelmien luokkiin, kun taas Shaker et al käsittelevät menetelmien käyttötapoja. Taulukossa 4 esitettyjen viiden menetelmäluokan lisäksi Hendrixk et al mainitsevat myös pseudosatunnaiset numerogeneraattorit (engl. pseudo-random number generators, PRNG) PCG:n generointimenetelmänä.

Taulukko 4. PCG-G:n menetelmien taksonomia Hendrixk et al (2013:1:8) perusteella.

Menetelmäluokka	Esimerkkimenetelmä
Generatiiviset kieliopit (engl. generative grammars)	Lindenmayer-systeemit
Kuvan suodatus (engl. image filtering)	Binäärinen morfologia (engl. binary morphology)
Spatiaaliset algoritmit (engl. spatial algorithms)	Fraktaalit
Monimutkaisten järjestelmien mallintaminen ja simulointi. (engl. modelling and simulation of complex systems)	Soluautomaatit (engl. cellular automata)
Tekoäly (engl. artificial intelligence)	Geneettiset algoritmit (engl. genetic algorithms)

2.5 Proseduraalisen sisällön generoinnin menetelmät

Shaker et al. (2016) laatimassa PCG:n tutkimuskentän katsauksessa on esitetty useita PCG:n menetelmiä ja niiden käyttökohteita. Näitä menetelmiä on kerätty taulukkoon 2, jossa esitetään niiden tiivistetyt selitykset ja generoitavat esimerkkisisällöt.

Taulukon 2 ensimmäisellä rivillä esitetty stokastinen haku- tai optimointialgoritmi edustaa hakupohjaisten menetelmien luokkaa. Kyseisen menetelmäluokka sisältää osanaan evolutionääriset algoritmit (engl. evolutionary algorithm, EA). (Shaker et al 2016:17) EA:t ovat yläluokka muun muassa geneettisille algoritmeille (GA), joita tarkastellaan lähemmin

niiden tutkimusten analyysin ja toteutustyön valossa.

Mainitussa GA:iden tutkimusten analyysissä keskitytään pelikenttiä generoiviin tutkimuksiin, minkä voidaan nähdä olevan yhteydessä (taulukon 5, rivi 4) soluautomaattien menetelmäluokkaan. Tämä perustuu myöhemmin käsiteltävään Pech et al. (2015) tutkimustyöhön, jossa soluautomaattien sääntöjä generoitiin geneettisen algoritmin avulla pelikenttien muodostamiseksi.

Taulukko 5. Proseduraalisen sisällön generoinnin menetelmiä Shaker et al. (2016) perusteella sekä niiden selityksiä ja esimerkkejä.

Generointimenetelmä	Tiivistetty selitys	Sisällön esimerkki
Stokastinen haku- tai optiomointialgoritmi	Sisältö muotoutuu iteratiivisesti: sisällön läpikäynti, muokkaaminen ja valitseminen (hyvien ominaisuuksien perusteella). (Shaker at al. 2016:17)	Pelitasot (Ashlock et al. 2011), tietokoneen ohjaamien pelaajien käyttäytyminen (Headleand et al. 2015) ja pelin säännöt (Shaker at al. 2016:99).
Tilan jakaminen	Jaetaan tilaa osajoukoiksi 2D tai 3D avaruudessa; tyypillisesti kyseessä on rekursiivinen toimintatapa. (Shaker at al. 2016:33)	Esitetty pelitasojen erillisten alueiden luominen (Shaker at al. 2016:34)
Ohjelmisto agentit	Esimerkiksi tekoälykäyttäytymisten (engl. artificial intelligence behaviors) hyödyntämistä sisällön generoinnissa. (Shaker et al. 2016:39)	Pelitasojen generoiminen (Shaker at al. 2016:38).
Soluautomaatit	Soluautomaatit (engl. cellular automata) koostuvat yksittäisistä automaateista, joita voidaan myös kuvata soluiksi (McIntosh 2010:36). Automaattitasolla käsittelyä voidaan jatkaa kahden tilajoukon ja n-ulotteisen ristikon esittämisellä, jotka toimivat automaatin osina. Tässä ensimmäinen joukko sisältää automaatin mahdollisia tilat (useita mahdollisuuksia) ja toinen joukko kattaa säännöt tilojen välisille siirtymille. Automaatin n-ulotteinen ristikko on yhteydessä sen naapuruston käsitteeseen, jolla määritellään kyseiseen automaattiin (tilat) vaikuttavat naapuriautomaatit. (Shaker et al. 2016:42).	Luolastot (Johnson et al. 2010) ja sokkelot (Pech et al. 2015).
Kieliopit	Merkkijonojen tuottamisen sääntöjä. Esimerkiksi kirjain 'A' korvataan kirjainyhdistelmällä 'AB' . (Shaker at al. 2016:74)	Kasvien ja pelitasojen generointi (Shaker at al. 2016:73).
Fraktaalit	Generoitava fraktaalimaasto (engl. fractal terrain) perustuu samana toistuviin piirteisiin, jotka ilmenevät yhä pienemmässä mittakaavassa (Shaker at al. 2016:62-63).	Maasto (engl. terrain) (Shaker at al. 2016:63).

Answer Set Programming, ASP	Rajoitteiden ja loogisten yhteyksien määrittelemistä logiikkaohjelmoinnin kautta. Rajoitteet täyttävää sisältöä muodostetaan ASP:n ratkaisijoiden (engl. solver) avulla. (Shaker at al. 2016:143)	Pelitasot (Shaker at al. 2016:143)
Satunnaisuus	Käytetään satunnaisuutta generoitavan sisällön määrittäjänä (Shaker at al. 2016:59).	Pelin maasto (Shaker at al. 2016:59)

2.6 Pelien proseduraalisen sisällön generoinnin menetelmiltä tavoiteltavat ominaisuudet

Shaker et al. (2016:6-7) ovat esittäneet, että PCG:n menetelmillä olisi seuraavat tavoiteltavat ominaisuudet: nopeus, luotettavuus, ohjattavuus (engl. controllability), ilmaisevuus (engl. expressivity), monimuotoisuus (engl. diversity) ja luovuus (engl. creativity) sekä uskottavuus (engl. believability).

Nopeudella tarkoitetaan generoinnin tarvitsemaa suoritusaikaa. Tätä voidaan käsitellä pelaamisen ja pelinkehityksen aikaisen generoinnin kannalta, joissa aikavaatimukset ovat erilaisia. (Shaker at al. 2016:6) Mahdollisimman nopea generointialgoritmi on oleellinen päämäärä, mutta generointitilanteen vaativuus toimii tässä tarkastelua ohjaavana tekijänä. Nopeustavoitteeseen pääsemistä voitaisiin helpottaa tuottamalla keskeinen sisältö ensin, mutta muun sisällön suhteen pitäisi pohtia esimerkiksi latausajan tai viiveellä ilmestyvän sisällön hyötyjä ja haittoja.

Luotettavuudella tarkoitetaan generoitavan sisällön laatua asetettujen kriteerien suhteen; tarkastelua suhteutetaan generoidun sisällön merkittävyyteen pelikokemuksen kannalta. (Shaker at al. 2016:6) Tämän ominaisuuden tavoittelemisessa voitaisiin hyödyntää priorisointia, jossa tärkeämmäksi määritelty sisältö tuotetaan suuremmilla resursseilla. Tässä oletuksena on, että resurssimäärällä on jokin oleellinen vaikutus generointiin ja niitä tulisi olla saatavilla riittävästi.

Ohjattavuus tarkoittaa generaattorin kykyä vastaanottaa lopputulokseen vaikuttavia käskyjä. Tässä ulkoinen osapuoli (ihminen tai algoritmi) määrittää joitakin sisällön ilmentäviä piirteitä. (Shaker at al. 2016:7) Tämän ominaisuuden tavoittelemisella voitaisiin olettaa olevan vaikutusta generointiaikaan; pelaajan toimiin perustuvassa generoinnissa

syötteen odottaminen voisi näkyä pidempänä generointiaikana. Tämä voisi puolestaan voisi näkyä pelaajaan tyytymättömyytenä. Generointialgoritmi voisi esimerkiksi jättää joitakin syötteitä pois, mikä voisi puolestaan näkyä sisällön heikompana luotettavuutena.

Ilmaisevuudessa yhtenä ääripäänä toimii sisällön yksittäisen ominaisuuden satunnainen vaihtelu ja toisena satunnaisesti koostetuista komponenteista koostuva sisältö. Monimuotoisuudella puolestaan tarkoitetaan sisältöä, joka ei toistu samankaltaisena. (Shaker at al. 2016:7) Ilmaisevuuden tavoittelemisessa voitaisiin käsitellä luotettavuutta; satunnaisista komponenteista koostuva, merkitykseltään korkea sisältö ei ole luotettavuudeltaan tasokasta. Monimuotoisuuden suhteen voitaisiin myös käsitellä laadukkuutta, mutta tässä arvioitaisiin sisällön merkitystä ja toistuvuuden vaikutuksia. Useasti samanlaisena toistuva, pelikokemuksen kannalta merkittävä sisältö ei välttämättä ole pelaajan kannalta laadukasta.

Luovuudella ja uskottavuudella tarkoitetaan sisällön keinotekoiselta näyttämisen välttämistä; pyritään sisältöön, joka ei vaikuta generaattorin suunnittelemalta. (Shaker et al. 2016:7) Näiden ominaisuuksien voidaan nähdä olevan yhteydessä kaikkiin edellä esitettyihin ominaisuuksiin, mutta ohjattavuudella voisi olla tässä erityisen suuri rooli. Pelaajan toimiin perustuva sisällön generoiminen voisi mahdollistaa keinotekoisuuden vaikutelman poistumisen, mutta tässä täytyisi huomioida pelaajan vaikutustapa sisällön generointiin.

3 GENEETTISET ALGORITMIT

Geneettiset algoritmit (GA:t) ovat luonnonvalinnan ja genetiikan mekanismeihin perustuvia hakualgoritmeja (engl. search algorithm) (Goldberg 1989:1). GA:t kuuluvat evoluutiolaskennan (engl. evolutionary computing, EC) tutkimusalueeseen (Eiben & Smith 2015:14), jossa ne ovat laajimmin tunnettu evolutionääristen algoritmien (engl. evolutionary algorithm, EA) tyyppi (Eiben & Smith 2015:99).

3.1 Geneettisten algoritmien tausta

1960-luvulla John Holland pyrki tutkimaan luonnossa tapahtuvaa sopeutumista ja kehittämään tapoja tämän ilmiön siirtämiseksi tietokonesysteemeihin; hän kehitti geneettiset algoritmit näiden pyrkimysten saavuttamiseksi. Holland jatkoi GA:iden kehittämistä Michiganin yliopistossa 1960 ja 1970-luvuilla oppilaidensa ja kollegoidensa avustamana. (Mitchell 1998:2-3) Goldbergin (1989:1) mukaan heillä oli kaksi tavoitetta: luonnon systeemien sopeutumisprosessien selittäminen ja abstrahoiminen sekä kyseisten systeemien mekanismeja ilmentävien ohjelmistojen suunnittelu.

Vuonna 1975 Holland esitti GA:n luonnossa tapahtuvan evoluution abstraktiona Adaptation in Natural and Artificial Systems –kirjassaan. Hänen GA:taan voidaan kuvata populaatioperustaiseksi algoritmiksi, jossa valinnan ja geneettisten operaattoreiden (rekombinaatio, mutaatio ja inversio) avulla siirrytään kromosomipopulaatiosta toiseen. (Holland 1992, Mitchell 1998:3 referoimana) Tämä algoritmi oli suuri innovaatio piirteittensä moninaisuuden osalta, mutta näiden alkuaikojen jälkeen GA:n määritelmä on siirtynyt kauemmas tästä alkuperäisestä muodosta (Mitchell 1998:3); GA:lle ei ole olemassa yleisesti hyväksyttyä ja samalla tiukkaa määritelmää. (Mitchell 1998:8).

Nykypäivänä GA:ita pidetään osana evoluutiolaskennan tutkimusaletta, jossa ne kuuluvat evolutionääristen algoritmien (EA) yläjoukkoon (Eiben & Smith 2015:14). Muita EA:hin kuuluvia algoritmijoukkoja (osa-alueita) ovat evoluutio-ohjelmointi (engl. evolutionary programming, EP), evoluutiostrategiat (engl. evolution strategies, ES) ja geneettinen ohjelmointi (engl. genetic programming, GP). (Eiben & Smith 2015:14)

3.2 Geneettiset algoritmit evoluution abstraktioina

Geneettiset algoritmit voidaan mieltää evoluution abstraktioiksi, jonka mahdollistajina toimivat valinta ja geneettiset operaattorit. (Holland 1992, Mitchell 1998:3 referoimana). Tässä alaluvussa käsitellään GA:iden biologiaan perustuvia ominaisuuksia. Käsittely jakautuu GA:iden ja biologian termien välisen suhteen määrittämiseen sekä GA-perustaisen evoluutioprosessin läpikäymiseen.

3.2.1 Biologian termit geneettisten algoritmien kannalta

GA:ssa esiintyvien biologian termien määritelmät on esitetty taulukossa 6 (tässä termien vastaavuudet ovat Mitchellin (1998) määritelmien mukaisia.) Taulukossa 7 tarjotaan vastaavanlainen tarkastelu geno- ja fenotyypin käsitteiden osalta (tässä määritelmät perustuvat Eibenin ja Smithin (2015) esityksiin).

GA:ssa käytettävät biologian termit kuvaavat suhteellisen yksinkertaisia kokonaisuuksia, kun taas niiden biologiset vastaavuudet ovat verrattain monimutkaisia (Mitchell 1998:5). Tämä ilmenee esimerkiksi taulukossa 6 esitetystä geenin määritelmästä, jossa luonnossa monimutkainen DNA-lohko esitetään GA:ssa (yksinkertaisimmillaan) yhtenä bittinä. Tämä ilmiö jatkuu kromosomitasolla; bittijonot toimivat DNA-ketjujen vastaavuuksina. GA:iden yksinkertaisilla muuttujilla ja tietorakenteilla voidaan nähdä olevan GA:iden laskennallista vaativuutta helpottava vaikutus, ja tämä yhtyy Goldbergin (1989:2) esittämään huomioon GA:iden laskennallisesta yksinkertaisuudesta.

Taulukossa 6 esitetty kromosomin määritelmä on samankaltainen taulukon 7 fenotyypin esitykseen verrattuna; molemmissa tapauksissa kyseessä on ongelman mahdollinen ratkaisu. Tässä tulee kiinnittää huomiota taulukon 7 mainintaan alkuperäisestä asiayhteydestä, minkä johdosta kyseessä on ongelman varsinainen ratkaisu käsiteltävän tietorakenteen sijasta.

Taulukko 6. Biologian termejä geneettisten algoritmien suhteen Mitchellin (1998:5-6) perusteella.

Termi	Määritelmä	
	Biologia	**Geneettiset algoritmit**
geeni	Proteiinia koodaava DNA-lohko.	Bitti tai bittijono.
alleeli	Geenimuoto.	Bittien tapauksessa 0 tai 1.
kromosomi	Geeneihin jaettava DNA-ketju.	Ongelman mahdollinen ratkaisu. Biteistä koostuva merkkijono (engl. bit string).
rekombinaatio	Kromosomien välinen geenien vaihtaminen uusien yksilöiden muodostamisen yhteydessä.	Yleensä yksikromosomisten vanhempien välistä bittien (geneettisen materiaalin) vaihtamista.
mutaatio	DNA:n alkeisosien muuttuminen vanhemmalta lapselle siirtyessä.	Geenin alleelin vaihtaminen satunnaisessa kohdassa bittien merkkijonoa (kromosomissa). Esimerkiksi 0-bitin vaihtaminen bitiksi 1.

Taulukon 7 Eibenin ja Smithin (2015) feno- ja genotyypin määritelmät liittyvät heidän esittämäänsä representaation (engl. representation) käsitteeseen, jolla on kaksi mahdollista merkitystä. Kyseessä on joko fenotyypin koodaaminen genotyypiksi tai genotyyppiavaruuden (engl. genotype space) tietorakenne. (Eiben & Smith 2015:29-30) Taulukon 7 määritelmät ovat samankaltaisia Mitchellin (1998:5-6) esitykseen verrattuna, mutta käsittelyä on tarkennettu ja viety lähemmäs GA:iden ja EA:iden nykytilaa Eibenin ja Smithin (2015) esityksien kautta.

Taulukko 7. Geno- ja fenotyypin määritelmät biologian sekä evolutionääristen algoritmien kannalta Eibening ja Smithin (2015:17-18,29) perusteella.

Termi	Määritelmä	
	Biologia	**Evolutionääriset algoritmit**
genotyyppi	Fenotyypin rakentamiseksi tarvittava informaatio (geenit).	Yksilöt EA:n sisällä. Fenotyypin koodaus (engl. encoding)
fenotyyppi	Genotyypin koodaama yksilön piirre. Esimerkiksi turkin väri.	Ratkaistavan ongelman mahdollinen ratkaisu alkuperäisessä asiayhteydessä.

3.2.2 Evoluutio evoluutiolaskennan näkökulmasta

Eiben ja Smith (2015) ovat esittäneet Darwinin evoluutioteorialle tiivistetyn muodon, jossa populaatio toimii evoluution yksikkönä (engl. unit of evolution) ja populaation muodostavat yksilöt ovat puolestaan valinnan yksikköjä (engl. unit of selection). Tässä valinta perustuu yksilön kykyyn sopeutua ympäristöönsä muihin yksilöihin suhteutettuna;

paremmin sopeutunut yksilö lisääntyy todennäköisemmin. Valittujen yksilöiden jälkeläisiin kohdistuvat mutaatiot mahdollistavat uusien yksilöiden muodostumisen ja testaamisen. Tämä johtaa populaation muuttumiseen, jolloin evoluutio jatkuu. (Eiben & Smith 2015:16)

Eiben ja Smith (2015:25) ovat myös esittäneet yleisen, kaikkien EA:iden perustan muodostavan rakenteen (engl. scheme). Heidän mukaansa EA rakentuu alustuksesta (engl. initalisation) ja lopettamisesta (engl. termination) sekä näiden välissä tapahtuvasta neljästä vaiheesta: vanhempien valinta (engl. parent selection), rekombinaatio, mutaatio ja selviäjien valinta (engl. survivor selection). Tässä populaatiosta valitaan vanhemmat, joiden kohdistetaan rekombinaatiota ja mutaatioita. Näiden operaattoreiden avulla muodostuvat jälkeläiset valitaan seuraavaan populaatioon. Heidän mukaansa evolutionääristen systeemien perustana toimii kaksi pääasiallista voimaa: variaatio (rekombinaatio ja mutaatio uusien ominaisuuksien mahdollistajina) ja valinta (ratkaisujen laadun keskiarvon kasvattaminen). (Eiben & Smith 2015:26-27).

Mitchellin (1998:4-5) mukaan evoluutio toimii etsintämenetelmänä kelpoisten yksilöiden löytämiseksi, jossa etsintä keskittyy (mahdollisiin) geneettisiin sekvensseihin. Hän on myös tarjonnut evoluution säännöille korkean tason esityksen, jossa lajien evoluutio perustuu satunnaisiin muutoksiin ja luonnonvalintaan. Tässä muutokset perustuvat esimerkiksi mutaatioihin, kun taas luonnonvalinta ilmenee kelpoisimpien yksilöiden selviämisenä ja lisääntymisenä; geneettinen materiaali siirtyy sukupolvelta toiselle. (Mitchell 1998:5)

3.2.3 Yksinkertainen geneettinen algoritmi

Yksinkertainen geneettinen algoritmi (SGA) on GA:iden alkuaikojen kehittämistyön tuloksena syntynyt GA:n määritelmä. Siinä käytetään binääristä representaatiota ja variaatio-operaattoreina toimivat rekombinaatio sekä mutaatio; rekombinaatiolla on verrattain suuri rooli uusien ratkaisujen luomisessa ja mutaation tapahtumisella on pieni todennäköisyys. SGA:ssa valinta perustuu kelpoisuuteen suhteutettuun valintaan (engl. fitness proportionate selection) ja jälkeläiset korvaavat vanhan populaation täysin. SGA:sta puuttuu monia GA:iden kehitystyön kautta löydettyjä ominaisuuksia. (Eiben & Smith

2015:99-100)

3.2.4 Yksinkertaisen geneettisen algoritmin työnkulku

Eiben ja Smith (2015:99-100) ovat esittäneet GA:n perinteisen työnkulun (engl. workflow) SGA:n käsittelemisen yhteydessä. Tässä he ovat käyttäneet seuraavia muuttujia:

μ: yksilöiden lukumäärä populaatiossa
p_c: kombinaation tapahtumisen todennäköisyys
p_m. yksittäisen bitin mutaatiotodennäköisyys

Esitetty työnkulku alkaa vanhempien valitsemisella, jossa valitaan μ vanhempaa väliaikaiseen populaatioon. Tästä ryhmästä valitaan satunnaisesti yksilöpareja kombinaatioon; parilla on mahdollisuus p_c tuottaa niitä korvaavia lapsia. Tämän jälkeen jokaiseen väliaikaisen populaation yksilöön kohdistetaan bittikohtaisia mutaatioita todennäköisyydellä p_m. Väliaikainen populaatio edustaa näiden vaiheiden jälkeen uutta sukupolvea. (Eiben & Smith 2015:99-100)

SGA:ssa vanhempien valitseminen toteutetaan kelpoisuuteen suhteutetusti (engl. fitness proportional selection, FPS) (Eiben & Smith 2015:100). Yksilön valinnan todennäköisyys riippuu yksilön itsensä ja muun populaation yksilöiden absoluuttisen kelpoisuuden suhteesta seuraavan funktion mukaisesti (Holland 1992, Eiben & Smith 2015:80 referoimana):

$$P_{FPS}(i) = \frac{f_i}{\sum_{j=1}^{\mu} f_j} \quad (1)$$

Tässä f_i on yksilön i absoluuttinen kelpoisuus populaatiossa, jossa on μ yksilöä. f_j on populaation yksilön j absoluuttinen kelpoisuus. (Eiben & Smith:79-80)

SGA:ssa mutaatio toimii luvun 3.2.1 taulukon 6 esityksen mukaisesti. Tässä yksittäisiä bittejä muokataan todennäköisyydellä p_m (Eiben & Smith 2015:100). SGA:n yhden pisteen kombinaatio alkaa kahden bittien merkkijonon (vanhemmat) kahtia jakamisella,

jossa merkkijonot jaetaan kahtia samasta kohdasta. Tämän kautta muodostuvat merkkijonojen loppuosiot vaihtavat keskenään paikkoja, mikä ilmenee kahden uuden yksilön muodostumisena (lapset). (Holland 1992, Eiben & Smith 2015:52-53 referoimana)

GA:n toimintaa voidaan kuvata Hollandin (1992) esittämän ja Mitchellin (1998:27) viittamaan rakennuspalikoiden (engl. building blocks) käsitteen avulla. Hollandin mukaan GA:iden toiminta perustuu hyvien rakennuspalikoiden löytämiseen, korostamiseen ja rekombinaatioon. Tässä hyvät rakennuspalikat toimivat hyvien ongelman ratkaisujen osina, ja niitä käsitellään rinnakkaisesti. (Holland 1992, Mitchell 1998:27 referoimana) Tämä rakennuspalikoiden käsite voidaan nähdä osana aiempaan SGA:n työnkulkua; valitut vanhemmat koostuvat hyvistä rakennuspalikoista ja ne siirtyvät näiden lapsiin rekombinaation kautta.

Eibenin ja Smithin (2015:99-100) mukaan SGA:sta on löydetty vikoja ja siitä myös puuttuu monia EA:ille kehitettyjä ominaisuuksia. Yhtenä tämän esimerkkinä toimii binääristen merkkijonojen korvaaminen jollakin muulla representaatiolla tarpeen vaatiessa. Eibenin ja Smithin huomiot toimivat perusteena myöhemmin esitettävän GA:n toteutustyön SGA:sta poikkeavien ominaisuuksien valitsemiselle.

4 GENEETTISTEN ALGORITMIEN KELPOISUUSFUNKTIOT SISÄLLÖN GENEROIMISESSA

Yannakakis ja Togelius (2011) ovat esittäneet kolme algoritmien kelpoisuusfunktioiden luokkaa. He kutsuivat näitä suoriksi arviointifunktioiksi (engl. direct evaluation function), simulaatioperustaisiksi arviointifunktioiksi (engl. simulation-based evaluation function) ja interaktiivisiksi arviointifunktioiksi (engl. interactive evaluation function).

Yannakakiksen ja Togeliuksen mukaan suorissa kelpoisuusfunktioissa sisällön laadun arvo (kelpoisuus) muodostuu suorasti sisällön ominaisuuksista. Tyypillisesti kyseessä on laskennallisesti helppo toimenpide, joka on räätälöity halutun pelin tiettyyn sisältöön. Suorat arviointifunktiot voidaan jakaa teoria- ja dataohjattujen funktioiden alaluokkiin. Teoriaohjatut funktiot perustuvat suunnittelijan tekemiin päätöksiin, jotka voivat pohjautua laadulliseen teoriaan (tunteisiin tai pelaajan kokemuksiin liittyen) tai suunnittelijan intuitioon. Tässä suunnittelija muodostaa kartoituksen kokemusmallin ja sisällön laadun välillä. Dataohjatut funktiot perustuvat datan keräämiseen sisällön vaikutuksen suhteen. Tätä voidaan toteuttaa esimerkiksi kyselyjen avulla. Tässä funktiojoukossa säädetään sisällön kartoittamista automaattisesti; sisältöä kartoitetaan pelaajan kokemuksiksi, jotka puolestaan kartoitetaan arviointifunktioiksi. (Yannakakis & Togelius 2011:154)

Yannakakiksen ja Togeliuksen mukaan simulaatioperustaisissa arviointifunktioissa sisällön laatu perustuu keinotekoisten agenttien käyttämiseen pelin pelaamisessa. Tässä pelattavuus toimii sisällön ominaisuuksien löytämisessä, jotka kaavoittuvat pelaajakokemusten malliin. Näitä malleja käytetään sisällön laadun arvon laskemiseen. Simulaatioperustaiset arviointifunktiot voidaan jakaa staattisiin ja dynaamisiin muotoihin. Staattisissa funktioissa peliä pelaava agentti ei muutu, kun taas dynaamisessa muodossa agentti muuttuu. (Yannakakis & Togelius 2011:154)

Yannakakiksen ja Togeliuksen mukaan interaktiivisissa arviointifunktioissa sisällön laatu perustuu pelaajan toimintaan. Pelaaja voi toimia datan lähteenä eksplisiittisesti tai implisiittisesti; pelaajalta voidaan kysyä dataa tai sitä voidaan kerätä pelaajan toimien perusteella. Tässä funktioluokassa data toimii pelaajan kokemusmallin räätälöijänä, mikä vaikuttaa arviointifunktioon. (Yannakakis & Togelius 2011:155)

5 GENEETTISET ALGORITMIT HAKUPERUSTAISEN SISÄLLÖN GENEROINNIN TUTKIMUKSESSA

Geneettiset algoritmit kuuluvat hakuperustaiseen proseduraalisen sisällön generointiin (SBPCG) evolutionääristen algoritmien alaluokkana (Shaker et al. 2016:17,19). SBPCG on PCG:n generoi ja testaa –toimintatavan erikoismuotona, joka kattaa kaikki heuristiset ja stokastiset haku- sekä optimointialgoritmit (Togelius et al. 2011:174).

Tässä luvussa käsitellään kahta PCG:n tutkimusta, joissa on generoitu sokkelonkaltaisia pelinkenttiä GA:n avulla. Tutkimusten etsintäprosessissa pyrittiin generoitavan sisällön samankaltaisuuteen ja se tehtiin Lappeenrannan tiedekirjaston tarjoaman LUT Finna-portaalin tietokantahakujen kautta.

Useiden PCG:n tutkimusten joukosta löytyi kaksi sokkelonkaltaisten pelitasojen GA-perustaisen generoinnin tutkimusta. Toisessa kyseisistä tutkimuksista pyrittiin vastaamaan ensimmäisenä julkaistun suoritusajan ongelmakohtaan. Tämän perusteella tutkimuksessa kehitettiin uusi GA, jota voitiin verrata aikaisempaan tutkimukseen erityisesti toteutuksen ja saavutusten osalta. Näissä tutkimuksissa sisällön fenotyyppi on samankaltaista, mutta genotyypissä ja sen koodaamisessa fenotyypiksi esiintyy vaihtelua.

Tässä luvussa on tarkoituksena tarjota käytännönläheisempi kuva GA:ista ja niiden toteuttamisesta PCG:n tutkimuksen kontekstissa. Tutkimukset esitellään lyhyesti, minkä jälkeen siirrytään niiden GA:iden komponenttien tarkastelemiseen alaluvuittain. Tutkimusten vertailemisella valotetaan toteutustapojen ominaisuuksia GA:iden teoriatiedon valossa.

5.1 Search-Based Procedural Generation of Maze-Like Levels -tutkimus

Ashlock et al. (2011) tutkivat sokkelonkaltaisten pelikarttojen generointia Search-Based Procedural Generation of Maze-Like Levels –tutkimuksessaan. Heidän päämääränään oli luoda generoitavan sokkelotyypin kontrolloimisen mahdollistava systeemi, jonka perustana toimisi evolutionäärinen algoritmi (Ashlock et al. 2011:260). Tämän yhteydessä he muun muassa tutkivat eri kelpoisuusfunktioiden käyttämistä useisiin sokkelojen representaatioihin (Ashlock et al. 2011:265). Kyseinen tutkimus oli jatkoa Ashlockin

(2010) Automatic Generation of Game Elements via Evolution –tutkimukselle, jossa hän tutki sokkeloiden generoimista shakin pelaamiseksi (Ashlock et al. 2011:260).

Ashlock et al. (2011) esittivät tutkimuksessaan neljä sokkelon representaatiota, joiden muodostama joukko jakautui kahtia suoriin ja epäsuoriin representaatioihin. Näihin representaatioihin perustuvien sokkeloiden kelpoisuutta arviointiin viidellä kelpoisuusfunktiolla, jotka pohjautuivat samassa yhteydessä kehitettyyn funktioiden suunnittelukehykseen. Kyseiset kelpoisuusfunktiot olivat dynaamisen ohjelmoinnin algoritmeja, joiden suunnittelemien ja toiminta perustui sokkeloihin asetettuihin tarkastuspisteisiin (engl. checkpoint). (Ashlock, et al. 2011:260, 265)

Ashlock et al. (2011) käyttivät tutkimuksessaan vakaan tilan (engl. steady state) GA:ta, jonka suorittaminen päätettiin 500 000 parittelutapahtuman jälkeen. He määrittelivät yhden sukupolven koostuvan 2000 parittelutapahtumasta, minkä johdosta jokaisella suorituskerralla käytetiin läpi 250 sukupolvea. GA:n suorittamista toistettiin koekerroittain 30 siemenluvulla. Populaatiokoko ei ollut sama koko tutkimuksessa: kolmessa neljästä representaatiosta käytettiin 120 yksilöä ja yhdessä neljästä 1000 yksilöä populaatiota kohden. (Ashlock et al. 2011:265)

Tässä tutkimuksessa kehitettiin populaation alustamistekniikka, jota Ashlock et al. (2011) kutsuivat harvan alustamisen (engl. sparse initialization) tekniikaksi. He pyrkivät tämän tekniikan avulla välttämään GA:n alkuperäisten sokkelopopulaatioiden kulkemattomuuden, joka perustui kuljettavien ja kulkemattomien sokkelon ruutujen yhtä suureen ilmenemistodennäköisyyteen (Ashlock, Lee et al. 2011) (s. 261-262). Harvassa alustamisessa käytetään fill-parametria alustavien sokkeloiden kuljettavuuden määrittämiseen, jossa (Ashlock et al. 2011:262):

$$0 < fill < 1 \qquad (2)$$

Tutkimuksen kokeissa fill asetettiin yleisesti arvoon 0,05. Tällä mahdollistettiin GA:n alustavan sokkelopopulaation yksilöiden kuljettavuus, ja näiden kelpoisuutta kasvatettiin variaatio ja valinta-operaattoreiden yhteisvaikutuksen avulla. Tässä variaatio mahdollisti

sokkelon esteiden lisäämisen, mikä ilmeni valintaperustaisesti kelpoisuuden kasvamisena. (Ashlock et al. 2011:262)

5.1.1 Representaatio

Tutkimuksessa esitettiin neljä sokkelonkaltaisen pelitason representaatiota, joista ensimmäinen suora muoto valitaan tämän alaluvun tarkentavaan käsittelyyn. Ashlock et al (2011) kutsuivat tätä ensimmäiseksi suoraksi representaatioksi (engl. first direct representation). (Ashlock et al. 2011:261) Kyseisen representaation valitseminen perustuu sen ja luvussa 3.2.3 esitetyn SGA:n yleisen genotyyppin samankaltaisuuteen; tällä valinnalla suoraviivaistetaan teoriatiedon käsittelemistä ja myöhempää tutkimusten vertailemista. Lisäksi epäsuorissa representaatioissa ilmenevät geno- ja fenotyypin välisen koodauksen (engl. encoding) erityishuomioita ei mielletä olennaisiksi GA:n toteuttamisen yleiskuvan viestimisen kannalta.

Valitussa suorassa representaatiossa GA:n fenotyyppinä toimi suorakulmio, jossa on liikkumisen mahdollistavia ja estäviä alueita. Genotyyppiä voidaan kuvata ruudukon jokaisen ruudun tilan koodaavaksi bittijonoksi. Tässä ruudulla liikkuminen on estetty, jos sitä koodaava bitti on yksi. (Ashlock et al. 2011:261-262).

5.1.2 Kelpoisuusfunktio

Tutkimuksessa esitettiin viisi kelpoisuusfunktiota (Ashlock et al. 2011:265), jotka olivat dynaamisen ohjelmoinnin algoritmeja (Ashlock et al. 2011:260). Niiden muodostamisessa hyödynnettiin 11 sokkelon ominaisuuksiin liittyvää määritelmää. Näistä kaksi seuraavaa ovat oleellisia tämän alaluvun tarkastelun kannalta (Ashlock et al. 2011:264-265):

- Sisäänkäynnin ruutua merkitään s-kirjaimella ja uloskäynnin ruutua e-kirjaimella.
- Minkä tahansa ruudun x ja sisäänkäyntiruudun s välisen reitin minimipituutta merkitään |x|. Jos etäisyyttä ei voida määrittää, niin |x|=-1.

Mainituista kelpoisuusfunktioista ensimmäinen huomioi sokkelon sisään- ja uloskäynnin välisen reitin pituuden, jossa pitkän reitin kelpoisuus on suurempi kuin lyhyen (Ashlock et al. 2011:266). Tämä kelpoisuusfunktio valitaan tarkempaan tarkasteluun, koska sen

suhteellinen yksinkertaisuus mahdollistaa GA:n oleellisten yleisominaisuuksien suoraviivaisen viestimisen ja sitä on myös hyödynnetty aikaisemmassa Ashlockin (2010) tutkimuksessa. Kyseistä funktiota kuvattiin seuraavasti (Ashlock et al. 2011:265):

$$F_1 = |e| \tag{3}$$

Merkintä $|e|$ on aikaisemmin esitettyjen määritelmien mukaisesti sokkelon uloskäyntireitin pituus. Ashlock et al.(2011:265) määrittelivät, että maksimoitava kelpoisuusfunktio saa kelpoisuuden nolla, jos seuraavat ehdot täyttyvät:

- Sisään- tai uloskäynnin ruudut ovat kulkemattomissa.
- Uloskäyntiruudulla e on vähemmän kuin k jäseniä, jossa k on pakollinen uloskäynnin tarkistuspisteiden jäsenyyden taso (engl. mandatory exit checkpoint membership level).

Muut tutkimuksen kelpoisuusfunktiot perustuivat muun muassa reitin tarkastuspisteiden kattamiseen ja sokkelon umpikujien (engl. cul-de-sac) määrään laskemiseen (Ashlock et al. 2011:265).

5.1.3 Populaatio

Tutkimuksessa ensimmäisen suoran representaation osalta käytettiin yleisesti populaatiokokoa 120. Lisäksi tutkijat testasivat suuren populaatiokoon ja edellä esitellyn harvan alustamisen tekniikan vaikutusta alustavien sokkeloiden kuljettavuuteen. (Ashlock et al. 2011:265)

5.1.4 Vanhempien valinta

Tutkimuksessa käytettiin yhden turnauksen valintaa, jossa ryhmäkoko oli seitsemän yksilöä. Näiden yksilöiden valitsemisessa ei käytetty yksilön korvaamista (engl. replacement). (Ashlock et al. 2011:265) Ashlockin (2006:35-36) esittämän käsittelyn perusteella yhden turnauksen valintaa voidaan kuvata seuraavan kaltaisena prosessina:

1. Sekoitetaan populaatio satunnaisesti.
2. Jaetaan populaatio pieniin ryhmiin.
3. Valitaan jokaisesta ryhmästä kaksi kelpoisinta yksilöä vanhemmiksi.
4. Luodaan vanhemmista variaatio-operaattoreiden avulla uusia yksilöjä (lapsia).
5. Korvataan ryhmän vähiten kelpoiset yksilöt uusilla.

5.1.5 Variaatio-operattorit

Tutkimuksen ensimmäisen suoran representaation yhteydessä käytettiin yhtenäistä crossoveria (engl. uniform crossover) ja mutaatiota (engl. uniform mutation). Kombinaation todennäköisyydeksi asetettiin 0,05 ja mutaatioille vastaava todennäköisyys oli 0,01. (Ashlock et al. 2011:261,265).

Yhtenäisessä crossoverissa lapsen jokainen geeni valitaan satunnaisesti yhdeltä vanhemmalta ja toinen lapsi luodaan tämän käänteisestä kuvauksesta. Geenien valitsemisessa hyödynnetään tasajakaumaa väliltä [0, 1], josta generoitu satunnaisluku määrää jokaisen geenin kohdalla vanhemman. Satunnaisluvun sijoittuminen määrätyn parametrin ympärille määrää valittavan vanhemman. (Eiben &, Smith 2015:53)

Eiben ja Smith (2015:57) ovat esittäneet GA:n reaalilukurepresentaatioiden käsittelemisen yhteydessä, että niissä tapahtuva yhtenäinen mutaatio on vastaavanlainen verrattuna binääristen koodausten bitin kääntämiseen (engl bit-flip).

5.2 Evolving Cellular Automata for Maze Generation -tutkimus

Pech et al. (2015) tutkivat sokkelonkaltaisten pelitasojen proseduraalista generointia Evolving Cellular Automata for Maze Generation –tutkimuksessaan. He käyttivät GA:ta toivottuja piirteitä ilmentävien sokkeloiden generoimiseen. GA:lla kehitettiin soluautomaatin (engl. cellular automaton) sääntöjä, joiden avulla mahdollistettiin ajonaikainen sokkelonkaltaisten tasojen generointi. (Pech et al. 2015:113)

Pechin et al. mukaan heidän esittämänsä lähestymistapa vastaa Ashlockin et al. (2011) (GA-perustainen generointi) ja Johnsonin et al. (2010) (CA-perustainen generointi) pelitasojen generoinnin tutkimusten ongelmakohtiin. He mainitsivat Ashlockin et al.

tutkimuksen suhteen, että siinä käytetty GA:n ei sovellu suoritusaikansa osalta sovelluksen reaaliaikaiseen generointiin. He kertoivat Johnson et al. tutkimuksen osalta, että CA:n sääntöjen manuaalinen suunnitteleminen voi olla hankalaa; tämä pätee erityisesti ominaisuuksiltaan toivottujen ja monimutkaisten tasojen tavoittelemiseen. (Pech et al. 2015:112-113)

Tutkimuksessa GA käsitteli soluautomaatin sääntöjä etsintätaulukkoina, jotka ottivat soluautomaatin naapuruston tulona ja muodostivat sen keskimmäiselle solulle uuden tilan (lähdön). Etsintaulukoille muodostettiin suora ja epäsuora representaatio, joissa lähtötilat (naapuruston keskimmäisen solun tila) esitettiin kokonaislukujen listana. (Pech et al. 2015: 116-117)

Kehitettyjen CA:n sääntöjen kelpoisuus perustui niiden avulla tuotettujen sokkeloiden toivottujen piirteiden täyttyvyyteen. Sokkeloja tuotettiin soluautomaateilla sadasta alustavasta sokkelokonfiguraatiosta (engl. initial maze configuration). Piirteiden täyttyvyyttä arvioitiin piirteen samanlaisuuden mitan (engl. attribute similarity measure, ASM) avulla. CA:n (sääntöjen) lopullinen kelpoisuusarvo muodostui kaikkien yksittäisellä soluautomaatilla generoidun pohjapiirrosten ASM-arvojen keskiarvosta. (Pech et al. 2015:118-119)

Pech et al. tutkivat lähestymistapansa tehokkuutta tasojen toivottujen piirteiden vaihtelemisen osalta. Lisäksi he selvittivät, millä parametreilla oli suuri vaikutus generoituihin sokkeloihin. Tutkimuksen kokeellisessa osuudessa käytettiin 12 yksittäisen kokeen neljään ryhmää. (Pech et al. 2015:119) Geneettistä algoritmia suoritettiin 30 kertaa jokaisessa kokeessa, minkä jälkeen viimeisen sukupolven paras kelpoisuus huomioitiin (Pech et al. 2015:121).

5.2.1 Representaatio

Tutkimuksessa esitettiin CA:n sääntöjen etsintätaulukolle suora ja epäsuora representaatio, joista suora muoto käsitellään tarkemmin tässä alaluvussa. Tällä valinnalla tuetaan luvussa myöhemmin esitettävää tutkimusten vertailemista; aiemmin luvussa 5.1.1 valittiin Ashlockin et al. (2011) tutkimuksesta suora representaatio tarkempaan tarkasteluun.

Suorassa representaatiossa etsintätaulukko kuvattiin listana, jossa kokonaislukuindeksin osoittama solun tila esitettiin binäärisesti; tila oli nolla tai yksi. Tässä listassa esitettiin tila jokaiselle mahdolliselle naapuruston konfiguraatiolle (etsintätaulukon indeksi). Soluautomaatin solun uuden tilan valitseminen perustui naapuruston tilojen binääriesityksen kymmenkantaiseen muotoon, joka toimi listan indeksinä. (Pech et al. 2015: 116-117)

Tutkimuksessa GA:lla kehitettiin CA:n sääntöjä, minkä johdosta genotyypin fenotyypiksi koodaaminen voidaan mieltää CA:n prosessiksi. Tässä CA hyödyntää tutkimuksessa mainittuja alustavia konfiguraatioita (Pech et al. 2015:118) (ruudukko) ja muodostaa uuden konfiguraation (fenotyyppi) kehitettyjen sääntöjen mukaisesti.

5.2.2 Kelpoisuusfunktio

Tutkimuksen kelpoisuusfunktio arvioi soluautomaatin (sääntöjen etsintätaulukko) kelpoisuutta piirteiden samanlaisuuden mitan (engl. attribute similarity measure, ASM) avulla; CA:lla tuotettujen tasojen ASM-arvojen keskiarvo on CA:n (sääntöjen) kelpoisuus. Tässä ASM-arvo laskettiin kaikille pohjapiirrosten tavoitelluille piirteille vertaamalla niitä piirroksista kuvankäsittelytekniikoilla eroteltuihin piirteisiin. ASM-arvojen selvittäminen mahdollistettiin käyttämällä kehitettyjä sääntöjä 100 alustavaan sokkelokonfiguraatioon. (Pech et al. 2015:118-119) Yksittäisen piirteen ASM-arvo laskettiin seuraavalla yhtälöllä (Pech et al. 2015:118):

$$ASM = \sum_{i=1}^{9} \left(1.0 - \frac{|da_i - aa_i|}{ma_i}\right)^3 \times aw_i \qquad (4)$$

Tässä aa kuvaa erotellun piirteen arvoa, ma on sen maksimiarvo ja da on piirteen tavoiteltu arvo. aw on piirteen painotuskerroin (engl. weighting factor), joiden summa kaikkien piirteiden osalta on yksi; tällä mahdollistetaan ASM-arvoille arvoväli [0.0, 1.0]. (Pech et al. 2015:119)

5.2.3 Vanhempien valinta

Tutkimuksessa vanhempien valinnassa hyödynnettiin elitismiä ja turnajaisvalintaa. Elitismi ilmeni viiden parhaimman yksilön suorana siirtämisenä vanhasta populaatiosta uuteen. Turnajaisvalinta tehtiin näiden yksilöiden siirtämisen jälkeen ja siinä huomioitiin kerralla 5 yksilöä. (Pech et al. 2015:199)

5.2.4 Variaatio-operaattorit

Tutkimuksessa käytettiin yhden pisteen crossoveria ja yhtenäistä mutaatiota. Nämä kohdistettiin turnajaisvalinnassa valittuihin vanhempiin ja näiden jälkeläisiin. (Pech et al. 2015:119)

Yhden pisteen crossover suoritettiin satunnaisesti ja siinä käytettiin kahta vanhempien kromosomia. Crossover mahdollistettiin ennen mutaatioiden tapahtumista. (Pech et al. 2015: 119) Yhtenäisessä mutaatiossa soluautomaatin sääntöjen etsintätaulukon listamuotojen yksittäisiä tiloja vaihdeltiin toisiin tiloihin satunnaisesti. (Pech et al. 2015:199)

5.3 Tutkimusten vertailu

Ashlockin et al. ja Pechin et al. tutkimukset vastaavat samanlaisen sisällön generointiongelmaan, mutta ne lähestyvät tämän GA-perustustaista ratkaisemista eri tavoin. Ashlock et al. käyttävät GA:ssaan heidän luomansa suunnittelukehyksen avulla tuotettuja kelpoisuusfunktioita. Pech et al. puolestaan hyödyntävät GA:ta CA:n sääntöjen generointiin, ja heidän kelpoisuusfunktionsa käsittelee sokkelotasolta toivottuja piirteitä.

Tässä luvussa käsitellään aluksi tutkimusten tavoitteita, tutkimusmenetelmiä ja saavutuksia. Näiden alalukujen jälkeen siirrytään tutkimusten GA:iden komponenttien vertaamiseen.

5.3.1 Tutkimusten tavoitteet

Ashlockin et al. (2011) mukaan heidän tutkimuksensa ensisijaiset panokset ovat sokkeloiden representaatiomahdollisuuksien tutkiminen ja kelpoisuusfunktioiden

suunnittelukehyksen tarjoaminen. Pechin et al. (2015) puolestaan ilmaisivat, että heidän panoksinaan toimivat GA:n ja CA:n yhdistäminen PCG:n kontekstissa, CA:n sääntötaulukon representaatioiden tutkiminen ja Ashlockin makukonseptin (engl. flavors) idean testaaminen.

Kummatkin tutkimukset ovat täten samankaltaisia representaatioiden tutkimispyrkimyksen osalta. Ashlock et al. kelpoisuusfunktioiden suunnittelukehys voidaan nähdä myöhemmän PCG:n kehitystyön mahdollistajana; Pech et al. (2015) ilmaisivatkin, että Ashlockin et al. tutkimuksen taustakartoituksessaan. He tosin keskittyivät pelitasojen GA-pohjaiseen kehittämiseen kelpoisuusfunktiokehyksen hyödyntämisen sijasta. Pech et al. myös jatkoivat SBPCG:n tutkimustyötä GA:iden, CA:iden ja PCG:n yhdistämisen kautta. Lisäksi he käsittelivät Ashlokin esittämää makukonseptia.

5.3.2 Tutkimusmenetelmät

Ashlock et al. (2011:265-266) tutkivat eri kelpoisuusfunktioiden käyttämistä neljässä representaatiossa ja sokkelon kokoluokan muuttamista. Lisäksi he suorittivat kokeita, joissa harvan alustamisen fill-parametrin arvoa korotettiin ja yhtenäinen crossoveri korvattiin yhden ja kahden pisteen crossoverilla. Pech et al. (2015:119-120) tutkivat tavoiteltavien sokkeloiden attribuuttiarvojen vaihtamista ja geneettisen algoritmin parametrien muuttamista. Geneettisen algoritmin muutettavia parametreja olivat representaation suoruuden ja epäsuoruuden vaihteleminen CA:n naapuruston säteen uudelleenasettamisen yhteydessä. Lisäksi CA:n solujen mahdollisten tilojen määrää ja CA:n suorittamisen toistoja muuteltiin. (Pech et al. 2015:120-121)

Ashlockin et al. ja Pechin et al. suorittamat tutkimukset ovat samankaltaisia representaation vaihtelemisen osalta; kummassakin tutkimuksessa tutkittiin suoraa ja epäsuoraa representaatioita tai representaatiojoukkoa. Pech et al. eivät tutkineet usean kelpoisuusfunktion vaikutusta generointiin, mutta he vaihtelevat kelpoisuusfunktion tavoiteltavia attribuutteja. Tämä yhdistää Ashlockin et al. ja Perchin et al. tutkimuksia kelpoisuustarkastelun muokkaamisen osalta. Pech et al. eivät vaihdelleen tutkimuksessaan valintaoperaattoreita, mikä toimii selvänä erona Ashlockin et al. tutkimukseen.

5.3.3 Tutkimusten saavutukset

Pechin et al. tutkimus voidaan nähdä jatkona Ashlockin et al tutkimukselle; Pech et al. esittelevät Ashlockin et al. työn oman tutkimuksensa taustatiedoissa ja pyrkivät ratkaisemaan siinä esiintyvän liian pitkän suoritusajan ongelman. Ashlock et al. tosin ilmaisivat tutkimuksessaan, että reaaliaikainen generointi voisi olla mahdollista algoritmin optimoinnin kautta. Pech et al. kuitenkin kuvasivat Ashlockin et al. algoritmin 20 minuutin suoritusaikaa liian pitkäksi tähän tehtävään.

Kummatkin tutkimukset onnistuvat esittämään sokkelonkaltaisten tasojen generointimenetelmän. Ashlockin et al. tutkimuksella on monia SBPCG:n GA-perustaisen muodon selvittämisen ansioita, mutta etenkin kelpoisuusfunktioiden suunnittelukehyksen toteuttamista voidaan pitää ansiokkaana tuotoksena. Pech et al. puolestaan esittelivät uuden tavan sokkelonkaltaisten tasojen nopeaan generointiin CA:iden mukaanottamisella GA:n generointiprosessiin (Pech et al. 2015).

5.3.4 Geneettisen algoritmin komponentit

Tässä alaluvussa käsitellään Ashlockin et al. ja Pechin et al. tutkimusten eroavaisuuksia toteutettujen geneettisten algoritmien komponenttien osalta. Taulukossa 8 on esitelty tutkimuksissa toteutettujen GA:iden komponentteja suorien representaatioiden osalta.

Suoran representaatiotyypin valitseminen rajaa komponenttikohtaista vertailua halittavaksi kokonaisuudeksi; tutkimuksissa esiintyi muun muassa epäsuoria representaatioita ja näihin liittyviä erityispiirteitä. Lisäksi Ashlockin et al. tutkimuksesta valittiin viidestä kelpoisuusfunktiosta yksi tarkempaan vertailuun, mikä perustuu kyseisen tutkimuksen kelpoisuusfunktioiden yleiseen samankaltaisuuteen dynaamisen ohjelmoinnin algoritmeina.

Taulukosta 8 voidaan huomata, että kummankin tutkimuksen genotyypeissä on samankaltaisuutta; kumpikin GA-toteutus käsittelee binäärilukujen sarjaa. Tämä yhtäläisyys ei kuitenkaan ulotu valitun tietorakenteen käyttämiseen: Ashlock et al. käyttävät bittijonoa sokkelon ruutujen kuljettavuuden määrittelemiseen, kun taas Pechin et al. listarakenne toimii CA:n sääntötaulukkona. Tätä voidaan kuvata genotyypin

dekoodaamisen eriäväisyydeksi, jossa Pechin et al. hyödyntämä CA tekee tästä verrattain epäsuoran prosessin Ashlockin et al. toteutukseen verrattuna.

Taulukko 8. SBPCG:n tutkimuksien GA:n komponentteja suorien representaatioiden osalta.

Tutkimus	**Ashlock et al. (2011): Search-Based Procedural Generation of Maze-Like Levels**	**Pech et al. (2015): Evolving Cellular Automata for Maze Generation**
Genotyyppi	Bittijono.	Lista binäärilukuja.
Fenotyyppi	Suorakulmio (matriisi), jossa liikkumisen sallivia ja estäviä ruutuja.	Ruudukko, jossa liikkumisen sallivia ja estäviä ruutuja.
Kelpoisuusfunktio	Dynaamisen ohjelmoinnin algoritmi, jossa maksimoidaan reitin pituus sokkelon alkupisteestä loppupisteeseen. Kelpoisuus on nolla, jos reitin kulkeminen on mahdotonta tai tarvittavat tarkistuspisteet eivät ole reitillä.	CA:lla tuotetun tason piirteiden samankaltaisuuden mitan (ASM) arvojen keskiarvo.
Vanhempien valinta	Yhden turnauksen valinta. Ryhmäkokona seitsemän yksilöä.	Elitismi ja turnajaivalinta viiden yksilön joukosta.
Variaatio-operaattorit	Yhtenäinen crossover ja yhtenäinen mutaatio.	Yhden pisteen crossover ja yhtenäinen mutaatio.

Taulukon 8 mukaan kummatkin tutkimukset käyttävät GA:n fenotyyppinä ruudukkomaista rakennetta, jossa on kulkemisen sallivia ja estäviä ruutuja. Tämä on tutkimuksia yhdistävä tekijä, ja tätä yhdistyvyyttä vahvistaa Pechin et al. tutkimuksen rooli Ashlockin et al. esittämän sokkelotasojen generoinnin idean jatkokehittäjinä.

Tutkimukset eroavat toisistaan merkittävästi käytetyn kelpoisuusfunktion osalta: Ashlock et al. käyttivät dynaamisen ohjelmoinnin algoritmia, jota voidaan pitää kaukaisena vaihtoehtona Pechin et al. käyttämään sokkelotason piirteiden mittaamiseen. Taulukon 8 tarkasteluun valittu dynaamisen ohjelmoinnin algoritmi huomioi sokkelotason piirteitä tarkistuspisteiden (engl. checkpoint) osalta, mutta näitä ei erotella sokkelosta kuvankäsittelytekniikoilla Pechin et al. tapaan. Kyseessä on tasolle ennalta määritelty tarkistuspisteiden joukko.

Taulukon 8 mukaan kummatkin tutkimukset käyttivät vanhempien valitsemisessa turnajaisvalintaa, mutta ryhmäkoossa ja turnajaisprosessissa esiintyi eriäväisyyksiä. Ashlock et al. käyttivät yhden turnauksen valintaa. Pech et al. hyödynsivät turnajaisvalintaa sen perusmuodossa.

Taulukon 8 mukaan tutkimusten variaatio-operaattorit ovat mutaation osalta samankaltaisia; mutaatiolla on genotyyppitietorakenteessa todennäköisyys yksittäisen bitin komplementointiin. Kombinaation osalta Ashlock et al. käyttivät pääasiassa yhtenäistä crossoveria, kun taas Pech et al. hyödynsivät yhden pisteen crossoveria. Ashlock et al. tosin kokeilivat myös yhden pisteen crossoveria harvan alustamisen tekniikan validoinnin yhteydessä. Yhden pisteen crossoveria voidaan pitää kombinaation toteuttamistapoja yhdistävä tekijänä, vaikka sillä ei ole merkittävää roolia Ashlockin et al. pääasiallisessa tutkimuksessa.

Ashlockin et al. ja Pechin et al. esittämillä GA:iden toteutuksilla on runsaasti eroja, mutta ne yhdistyvät lopulta toisiinsa samankaltaisen sisällön generoinnin suhteen. Pechin et al. lähestymistapa, jossa he käyttävät GA:ta CA:n sääntötaulukkojen kehittämiseen on yksi merkittävistä tekijöistä näille eroavaisuuksille. Sääntötaulukoiden generoiminen tekee genotyypin dekoodaamisesta verrattain epäsuoran prosessin, ja CA:n mahdollistamien rakenteiden arviointi vaatii CA:n suorittamisen.

6 GENEETTISEN ALGORITMIN TOTEUTTAMINEN GO-OHJELMOINTIKIELELLÄ

Tämä luku käsittelee SBPCG:tä GA:n toteuttamistyön analyysin kautta. Toteutetun GA:n avulla luodaan 2-ulotteisia sokkelomaisia pelikenttiä. Nämä kentät ovat matriisirakenteita, joissa yksittäiset solut estävät tai sallivat niissä liikkumisen vaaka- tai pystysuunnassa.

Sokkeloita generoivan GA:n toteuttaminen toimii yhdistävänä tekijänä aiemmin läpikäytyihin Ashlockin et al. (2011) ja Pechin et al (2015) tutkimuksiin, joissa generoitiin samankaltaisia 2-ulotteisia sokkeloita. Aiemmin esitetty tutkimuksien käsitteleminen ja vertailu toimivat perustana tämän toteutustyön tekemiselle. Generoitavan sisällön luoteen lisäksi tämä näkyy Ashlockin et al. (2011) esittämän harvan alustamisen parametrin arvojen vaihtelemisen tutkimisena. Tarkoituksena on tarjota käytännönläheisempi näkemys GA-perustaiseen sisällön generointiin, jossa aiemmin esitetty tutkimustieto helpottaa toteutustyön kokonaisuuden omaksumista.

6.1 Geneettisen algoritmin komponenttien valitseminen

Eiben ja Smith (2015:28) ovat esittäneet evolutionääristen algoritmien yleiset komponentit, jotka toimivat tämän toteutustyön perustana. Taulukossa 9 on esitetty GA:n komponenttivalinnat, jotka poikkeavat erityisesti genotyypin osalta yksinkertaisen GA:n määritelmästä. Muita huomionarvoisia komponenttivalintoja ovat leveyshakualgoritmin hyödyntäminen kelpoisuusfunktiona ja turnajaisvalinnan käyttäminen vanhempien valitsemiseen populaatiosta.

Genotyypin valitsemisessa päädyttiin boolean-muuttujien hyödyntämiseen binäärilukujen sijasta. Tämä mahdollisti ohjelmallisen toteutuksen yksinkertaistamisen muun muassa genotyypin mutaatioiden suhteen, jotka toteutetaan nyt boolean-muuttujien komplementointina.

Leveyshakualgoritmin hyödyntämisellä sokkeloiden kelpoisuuden laskemisessa pyrittiin kelpoisuusfunktion yksinkertaisuuteen. Leveyshaku voidaan myös sovittaa matriisirakenteiden lyhyimpien reittien löytämiseen. Kyseisten reittien maksimoimisella pyritään yhä pidempiin reitteihin sokkelon läpi; reitin pituus on suoraan verrannollinen

sokkelon kelpoisuuteen. Leveyshakuun perustuvaa kelpoisuusfunktiota voidaan pitää luvussa 4 esitellyn Togeliuksen ja Yannakakiksen (2011) määrittelemän dataohjatun suoran kelpoisuusfunktion ilmentymänä. Tämä perustuu käytettävän leveyshaun ja sen suorittamisen tuloksen suoraan vaikutukseen sisällön kelpoisuuden määrittämisessä.

Turnajaisvalinnan käyttämisellä pyrittiin vanhempien valinnan mekanismin yksinkertaisuuteen. Tässä jokainen vanhempi valitaan rajatusta populaation alajoukosta kelpoisuusarvoja vertailemalla. Tämä johtaa lopuksi yhden kelpoisimman tai ryhmän ensimmäisen yksilön valitsemiseen, jos kaikki ryhmän jäsenet ovat yhtä kelpoisia.

Taulukko 9. Toteutettavan GA:n komponenttivalinnat.

Komponentti	Toteutustapa
representaatio (geno- ja fenotyyppi) (engl. representation)	Genotyyppinä 1-ulotteinen vektori. Fenotyyppinä 2-ulotteinen vektori, jossa on kuljettavia ja kulkemattomia soluja. Solut ovat tietueita, joille on määritelty kuljettavuus, etäisyys sokkelon alkupisteeseen ja väri.
kelpoisuusfunktio (engl. fitness function)	Leveyshaku (engl. breadth-first search) lyhyimmän reitin löytämiseen 2-ulotteisesta vektorista. Tämän reitin pituutta maksimoidaan ja sen pituus on suoraan verrannollinen sokkelon kelpoisuuteen.
populaatio (engl. population)	100 yksilöä.
vanhempien valitseminen (engl. parent selection)	Turnajaisvalinta (engl. tournament selection), jossa valitaan satunnaisesti asetettu määrä yksilöitä turnajaiseen. Sama yksilö voidaan valita useasti ja turnajaiset johtavat yhden vanhemman valitsemiseen.
variaatio-operaattorit (engl. variation operators)	Mutaatiossa boolean-muuttujan komplementointi, jossa yhden alleelin mutaatiotodennäköisyys on 0.05. Kombinaatiossa yhden pisteen kombinaatio (engl. single point combination), jossa kromosomipari (vanhemmat) vaihtavat osiansa yhden valitun pisteen perusteella (piste ei voi sijaita kromosomien alku- tai loppulokuksessa) todennäköisyydellä 0.95.
selviytyjien valitseminen (engl. survivor selection)	Vanha populaatio korvataan täysin uusilla jäsenillä ilman erillistä valintaa.

6.2 Toteutustyön vaiheet

Toteutustyön suorittaminen alkoi tarvittavien GA:n komponenttien suhteuttamisella ratkaistavaan sokkeloiden generoinnin ongelmaan. Tämän jälkeen tarkasteltiin projektin osittamista paketteihin ja niiden sisällä funktioihin Go-ohjelmointikielellä.

Hyödynnettävä geneettinen algoritmi sijoitettiin ohjelman suorituksen kontrollista erilliseen pakettiin, jossa osittamista jatkettiin tiedostotasolla. Geneettisen algoritmin genotyyppi ja siihen läheisesti liittyvät funktiot (erityisesti mutaatio, rekombinaatio ja fenotyypiksi muuntaminen) sijoitettiin omaan tiedostoonsa. Geneettisen algoritmin abstraktimmat vaiheet, kuten variaation ja valintaoperaattoreiden hallinnan funktiot erotettiin geno- ja fenotyypin muodostamista kokonaisuuksista omaan osuuteensa. Tällä pyrittiin generointialgoritmin kontrollin ja ohjattavien osien erottamiseen.

Toteutustyön suurimpana haasteena toimi leveyshakualgoritmin sovittaminen sokkeloiden kelpoisuuden arvioimiseen. Tässä päädyttiin hyödyntämään solua kuvaavaa tietuetta (engl. struct); jokainen sokkelon piste on yksi solutietue.

Soluille määriteltiin seuraavat ominaisuudet: kuljettavuus, etäisyys sokkelon alkupisteeseen ja väri. Väriä käytettiin sokkeloiden kuvien muodostamisessa, jossa kuljettavuus merkittiin valkoisella ja tämän estyminen mustalla. Sokkeloiden lyhyintä reittiä merkittiin sinisillä soluilla.

6.3 sisällön generoinnin tulokset

Sokkeloiden generoinnin osalta suoritettiin kaksi koetta, joissa molemmissa kokeiltiin Ashlockin et al. (2011) esittämän harvan alustamisen parametrin vaihtelemista. Tarkemmin näissä kokeissa vaihdeltiin alustavien sokkeloiden yksittäisten solujen todennäköisyyttä olla kuljettavissa.

Ensimmäisessä kokeessa generointiin 16x16-sokkeloita aiemmin mainitun alustamistodennäköisyyden arvoilla 0.5 ja 0.75. Näiden sokkeloiden generointiaika oli varsin lyhyt (alle minuutin), ja niiden generointi sopisivat luultavasti luvun 2.4 taulukon 3 ulottuvuuden 1 online-ääripäähän (sisältöä generoidaan pelin pelaamisen aikana).

Toisessa kokeessa generointiin 64x64-sokkeloita niiden ulkonäön katselmoimiseksi ja samalla vaihdeltiin alustamistodennäköisyyttä välillä [0.5;0.75]. Näiden sokkeloiden generointiaika laskettiin minuuteissa, minkä johdosta niiden generointi kuuluu todennäköisesti luvun 2.4 taulukon 3 ulottuvuuden 1 offline-ääripäähän (sisällön generointi pelikehityksen aikana).

Kumpaakin esiteltyyn sokkeloluokkaan (16x16 ja 64x64) sisältyvät sokkelot voisivat toimia lisäsisältönä peleissä, joissa sokkeloiden ratkaiseminen on pelin lisäominaisuus tai -haaste. Näiden sokkeloluokkien hyödyntäminen pelin ensisijaisessa sisällössä vaatii enemmän tutkimustyötä tulosten laadun varmistamiseksi. Tämä pätee erityisesti pelin pelaamisen aikana tapahtuvaan generointiin.

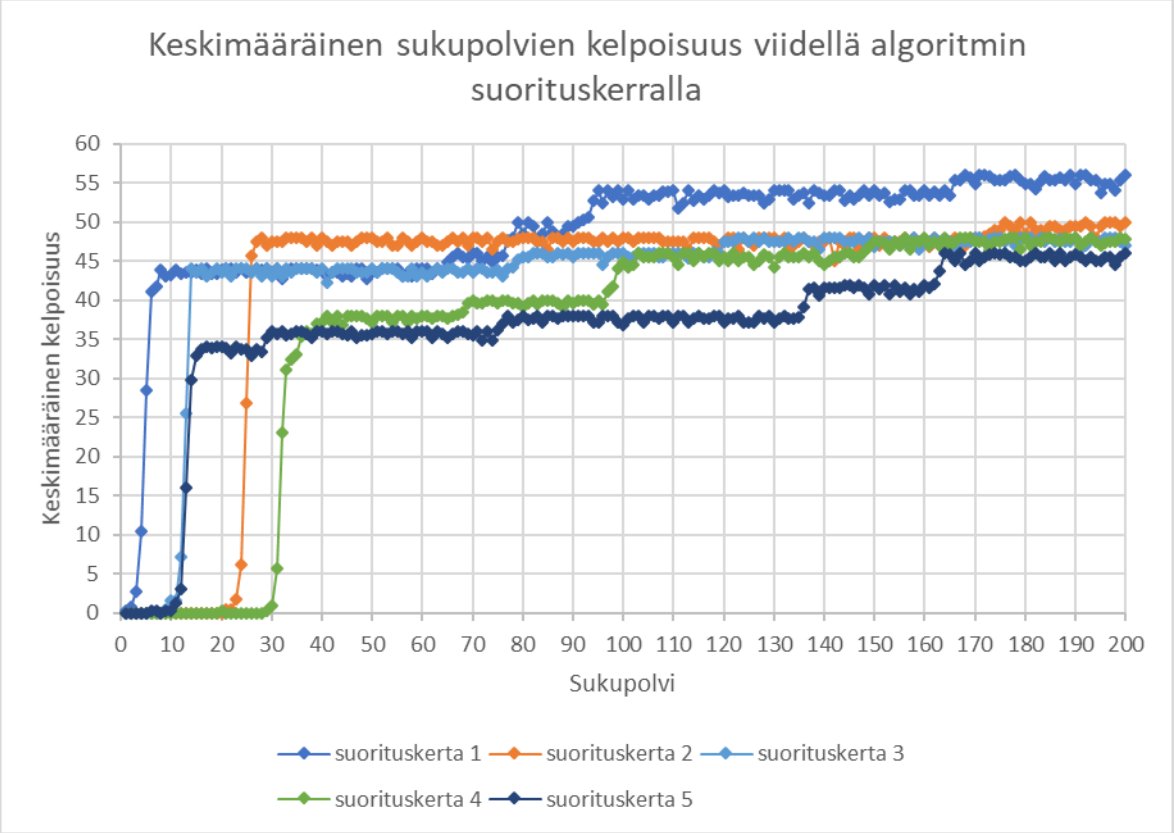
6.3.1 16x16-sokkeloiden populaatioiden harvan alustamisen parametrin vaihtamisen tutkiminen

Tässä kokeessa GA:lle asetettiin taulukossa 10 esitetyt parametrit. Alustamistodennäköisyyden arvoa 0.5 käyttämällä saatiin aikaan kuvaajan 1 mukaiset sukupolvien keskimääräiset kelpoisuudet viidelle algoritmin suorituskerralle.

Taulukko 10. Ensimmäisen kokeen geneettisen algoritmin parametrit.

Sokkeloiden koko	Populaatio	Sukupolvien lukumäärä	Turnajaisvalinnan ryhmäkoko	Rekombinaation %	Mutaation %	Harvan alustamisen %
16x16	100	200	8	95	5	50 ja 75

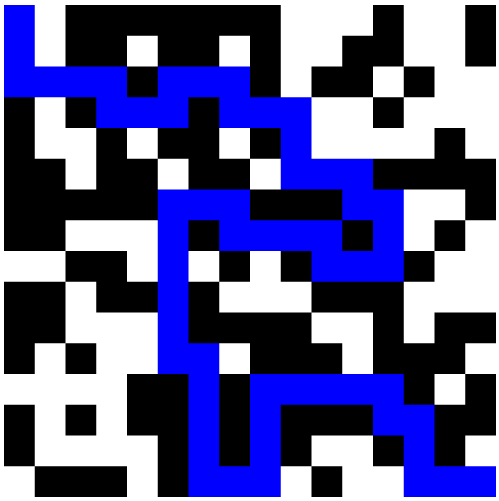
Kuvaajasta 1 voidaan huomata suorituskertojen 1 ja 4 välinen suuri ero, jonka voidaan olettaa johtuvan satunnaisesta vaihtelusta populaatioiden luomisen yhteydessä. Lisäksi voidaan todeta, että ensimmäisten populaatioiden aikana tapahtuu nopeaa keskimääräisen kelpoisuuden kasvua. Tämä kasvu myös laantuu nopeasti, ja tämän jälkeen sukupolvissa tapahtuu satunnaista kelpoisuuden kasvua suhteelliselta kelpoisuudeltaan hyvien yksilöiden yleistyessä. Sukupolvien 165-200 aikana keskimääräisen kelpoisuuden voidaan nähdä pysyvän samana, mutta tämän jälkeen tapahtuvaa kehitystä on vaikea ennustaa GA:n satunnaisuuteen perustuvien ominaisuuksien johdosta.



Kuvaaja 1. Keskimääräinen sukupolvien kelpoisuus viidellä algoritmin suorituskerralla harvan alustamisen todennäköisyyden ollessa 0.5.

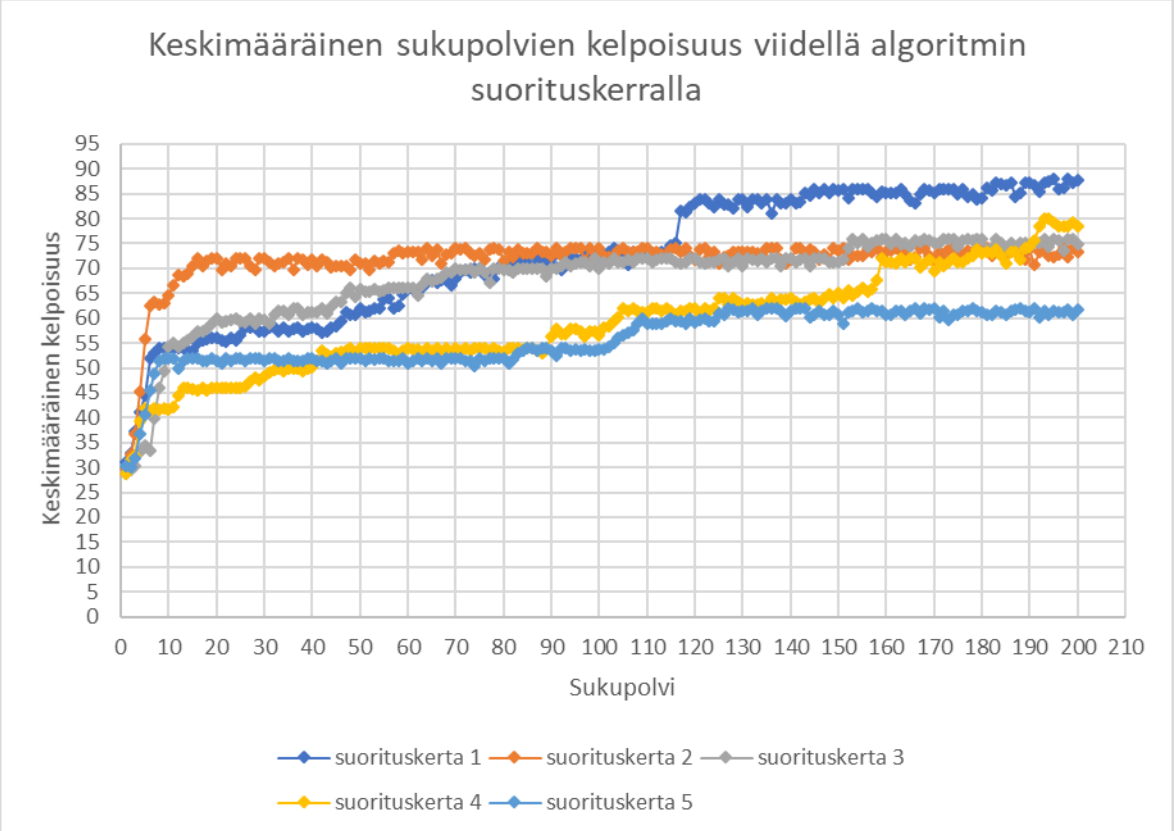
Kuvassa 1 on esitetty yksi testin 1 viimeisen sukupolven sokkeloista. Tässä siniset solut merkitsevät lyhintä löydettyä reitti sokkelon alkupisteestä (vasen yläreuna) loppupisteeseen (oikea alareuna). Kyseessä on 16x16-sokkelo, josta voidaan huomata sokkelon läpi kulkevan reitin koukisteleva luonne.

Kuvaajassa 2 on esitetty viisi algoritmin suorituskertaa harvan alustamisen todennäköisyydellä 0.75. Siitä voidaan todeta, että suurempi harvan alustamisen todennäköisyys näyttää parantavan sokkeloiden ensimmäisten sukupolvien keskimääräistä kelpoisuutta. Tässä kuvaajassa ensimmäisten sukupolvien aikana tapahtuu kuvaajan 1 testien tapaan nopeata kasvua, ja kelpoisuuden kasvaminen tasaantuu samaan tapaan myöhemmissä sukupolvissa. Kuvaajan 2 esittämissä suorituskerroilla kelpoisuusarvot ovat korkeampia kuvaajan 1 vastaaviin verrattuna ja heikoimmin kehittynyt populaatio (suorituskerta 5) saavuttaa kuvaajan 1 parhaimman populaation (suorituskerta 1) lopullisen keskimääräisen kelpoisuuden.

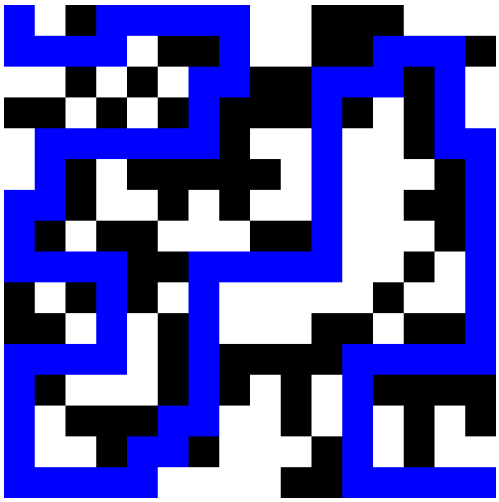


Kuva 1. Yksi kuvaajan 1 suorituskerran 1 viimeisen sukupolven sokkeloista. Valkoiset ja siniset ruudut ovat kuljettavissa. Sininen väri merkitsee lyhintä sokkelon reittiä ja mustalla merkitään kulkemattomissa olevia ruutuja. Tämän sokkelon kelpoisuus (sinisen reitin pituus ruutuina) on 56 .

Kuvassa 2 on esitetty yksi kuvaajan 2 suorituskerran 1 viimeisen sukupolven sokkeloista. Kyseessä on 16x16 sokkelo, jossa sinisellä merkitty reitti sokkelon läpi on kuvan 1 sokkelon reittiin verrattuna pidempi.



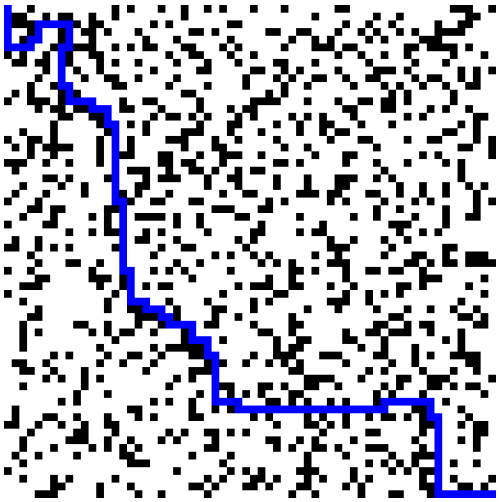
Kuvaaja 2. Keskimääräinen sukupolvien kelpoisuus viidellä algoritmin suorituskerralla harvan alustamisen todennäköisyyden ollessa 0.75.



Kuva 2. Yksi kuvaajan 2 suorituskerran 1 viimeisen sukupolven sokkeloista. Valkoiset ja siniset ruudut ovat kuljettavissa. Sininen väri merkitsee lyhintä sokkelon reittiä ja mustalla merkitään kulkemattomissa olevia ruutuja. Sokkelon kelpoisuus (sinisen reitin pituus) on 88.

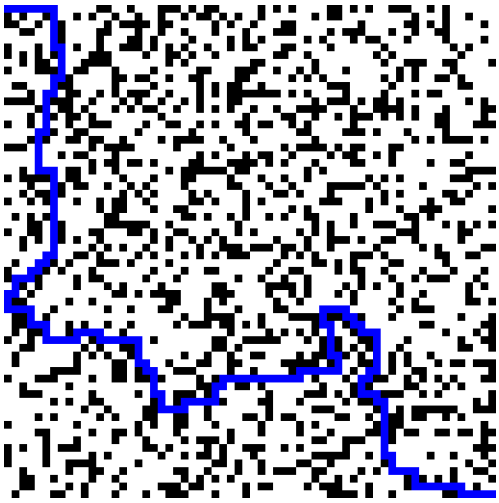
6.3.2 64x64 sokkeloiden harvan alustamisen parametrin vaihteleminen ja generoitujen sokkeloiden ulkonäön katselmointi

Tällä kokeella pyrittiin ilmaisemaan harvan alustamisen vaikutusta suurempien sokkeloiden ulkoasuun. Kuvassa 3 on esitetty 64x64-sokkelo, jonka generoinnissa käytettiin harvan alustamisen todennäköisyyttä 0.75.



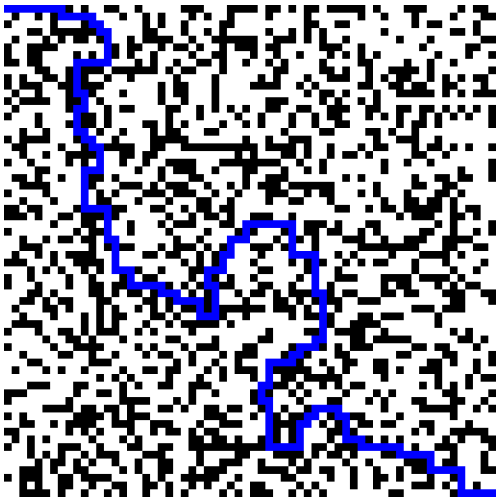
Kuva 3. Harvan alustamisen arvolla 0.75 generoitu 64x64-sokkelo, jonka kelpoisuusarvo (sinisen reitin pituus ruutuina) on 136.

Kuvasta voidaan todeta, että löydetty reitti vaikuttaa yksinkertaiselta; se etenee varsin suorasti sokkelon alkupisteestä loppupisteeseen. Kuvassa 4 on esitetty hieman mutkittelevamman reitin omaava sokkelo, jonka generoimisessa käytettiin harvan alustamisen todennäköisyyttä 0.7.

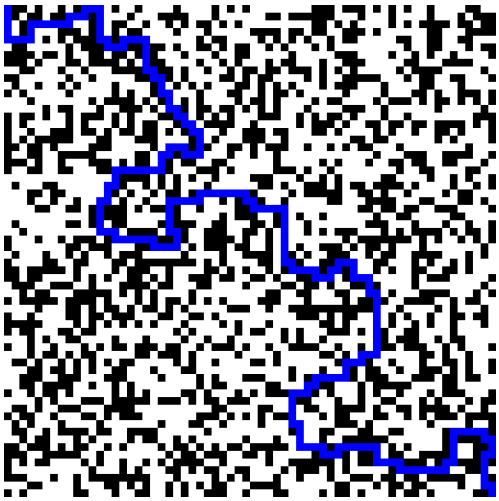


Kuva 4. Harvan alustamisen arvolla 0.7 generoitu 64x64-sokkelo, jonka kelpoisuusarvo (sinisen reitin pituus ruutuina) on 180.

Kuvassa 5 on esitetty sokkelo, jossa käytettiin harvan alustamisen todennäköisyyttä 0.65 ja kuvan 6 sokkelon generoimisessa vastaava todennäköisyys oli 0.6. Näissä sokkeloissa on huomattavissa mutkitteleva reitti, joka pyrkii seuraamaan sokkelon diagonaalia.



Kuva 5. Harvan alustamisen arvolla 0.65 generoitu 64x64-sokkelo, jonka kelpoisuusarvo (sinisen reitin pituus ruutuina) on 192.



Kuva 6. Harvan alustamisen arvolla 0.6 generoitu 64x64-sokkelo, jonka kelpoisuusarvo (sinisen reitin pituus ruutuina) on 220.

Esitettyjen alustamistodennäköisyyksien lisäksi kokeiltiin todennäköisyyksiä 0.55 ja 0.5. Näillä arvoilla generoidut sokkelot olivat kelvottomia; niiden kelpoisuus oli nolla ja täten niiden läpi ei kulkenut yhtäkään reittiä.

7 POHDINTA, TULEVAISUUS JA YHTEENVETO

Tässä työssä esitetty PCG:n teoriakatsaus viestii sen potentiaalista peliteollisuuden tulevaisuuden ongelmien ratkaisemisessa ja pelikehittähien toimintakyvyn tehostajana. PCG:n ja GA:iden SBPCG:hen yhdistämisen kautta on voitu saada viitteitä PCG monimuotoisuudesta ja kytkentämahdollisuuksista monien vaikeiden ongelmien ratkaisemiseen. Tämän työn alussa esitetty PCG:n teoriapohja toimii esimerkkinä sisällön generoinnin varteenotettavuudesta tieteellisen kehittämisen ja käytännön näkökulmista.

Työssä esitetty GA:n toteutustyö toimi luonnollisena jatkeena SBPCG:n tutkimusten käsittelemiselle ja sen avulla pystyttiin viestimään PCG:n helposta lähestyttävyydestä hakuongelmien ratkaisemisessa. Toteutettu geneettinen algoritmi mahdollistaa yksinkertaisten sokkeloiden generoimisen, mutta sitä voitaisiin vielä kehittää toiminnan kattavuuden suhteen. Nykyinen toteutus generoi sokkeloita tavalla, jossa ainoastaan lyhyimmällä reitillä on merkitystä sokkelon kelpoisuuden laskemiseksi. Generoitavista sokkeloista tulisi luultavasti mielenkiintoisempia, jos kelpoisuusfunktio arvioisi monia pitkiä reittejä sisältävät sokkelot kelpoisimmiksi yksinkertaisempiin sokkeloihin verrattuna.

GA:n ohjelmallista toteutusta voitaisiin kehittää ottamalla Go-ohjelmointikielen tarjoama rinnakkainen suoritus täyteen käyttöön. Nykyisessä versiossa geneettinen algoritmi on rinnakkainen ajettava rutiini, joka viestii pääohjelmalle tilannepäivityksiä kanavien (engl. channel) välityksellä. Tulevaisuudessa pääohjelma voisi luoda useita geneettisen algoritmin ilmentymiä, jotka toimisivat rinnakkaisesti. Näiden algoritmien sisällä rinnakkaisjakoa voitaisiin jatkaa esimerkiksi kelpoisuusfunktion suorittamien laskutoimitusten rinnakkaisena toteuttamisella. Tällä tavalla voitaisiin hyödyntää moniydinprosessorien tarjoamia mahdollisuuksia ja täten lyhentää GA:n suoritusaikaa. Nykyinen GA voitaisiin myös asettaa palvelinympäristöön Go-ohjelmointikielen tarjoaman valmiin toimintakontekstin avulla, mikä voisi mahdollistaa GA:n hyödyntämispotentiaalin kasvamisen.

LÄHTEET

Ashlock D, Lee C, McGuinness C. Search-Based Procedural Generation of Maze-Like Levels. *IEEE Transactions on Computational Intelligence and AI in Games* 2011;3(3):260-273.

Ashlock, D., 2006. *Evolutionary Computation for Modeling and Optimization*. New York, NY, USA: Springer Science+Business Media.

Ashlock, D., 2010. Automatic generation of game elements via evolution. *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, Aug 2010, pp. 289-296.

Eiben, A.E. and Smith, J.E., 2015. *Introduction to Evolutionary Computing*. 2nd ed. 2015 edn. Berlin, Heidelberg: Springer Berlin Heidelberg.

Goldberg, D.E., 1989. *Genetic algorithms in search, optimization, and machine learning*. Repr. with corr edn. Reading, Mass: Addison-Wesley.

Headleand, C.J., Henshall, G., Cenydd, L.A. and Teahan, W.J., 2015. Towards Real-Time Behavioral Evolution in Video Games. In: C.J., Teahan, W.J., Ap Cenydd, L., eds, *Artificial Life and Intelligent Agents: First International Symposium, ALIA 2014, Bangor, UK, November 5-6, 2014. Revised Selected Paper*s. Cham: Springer International Publishing, pp. 3-16.

Hendrikx, M., Meijer, S., Van Der Velden, J. and Iosup, A., 2013. Procedural Content Generation for Games: A Survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 9(1), pp. 1-22.

Holland, J.H., 1992. *Adaptation in Natural and Artificial Systems*. 1. ed. edn. Cambridge, Mass. u.a: MIT Press.

Johnson, L., Yannakakis, G.N. and Togelius, J., 2010. Cellular Automata for Real-time Generation of Infinite Cave Levels, *Proceedings of the 2010 Workshop on Procedural Content Generation in Games* 2010, ACM, pp. 10:4.

Mahlmann, T., Togelius, J. and Yannakakis, G.N., 2012. Spicing Up Map Generation. In: Di Chio, C., Agapitos, A., Cagnoni, S., Cotta, C., de Vega, F.F., Di Caro, G.A., Drechsler, R., Ekart, A., Esparcia-Alcazar, A.I., Farooq, M., Langdon, W.B., Merelo-Guervos, J.J., Preuss, M., Richter, H., Silva, S., Simoes, A., Squillero, G., Tarantino, E., Tettamanzi, A.G.B, Togelius, J., Urquhart, N., Uyar, A. S., Yannakakis, G.N., eds, *Applications of Evolutionary Computation: EvoApplications 2012: EvoCOMNET, EvoCOMPLEX, EvoFIN, EvoGAMES, EvoHOT, EvoIASP, EvoNUM, EvoPAR, EvoRISK, EvoSTIM, and EvoSTOC, Malaga, Spain, April 11-13, 2012, Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 224-233.

McIntosh HV. Life's Still Lifes. In: Adamatzky A, ed. *Game of Life Cellular Automata* London: Springer London; 2010. p. 35-50.

Mitchell, M., 1998. *An Introduction to Genetic Algorithms*. 7. print. edn. Cambridge, Massachusetts: MIT Press.

Pech A, Hingston P, Masek M, Lam CP. Evolving Cellular Automata for Maze Generation. In: Chalup, S.K., Blair, A.D., Randall, M., eds. *Artificial Life and Computational Intelligence: First Australasian Conference, ACALCI 2015, Newcastle, NSW, Australia, February 5-7, 2015*. Proceedings Cham: Springer International Publishing; 2015. p. 112-124.

Shaker, N., Togelius, J., and Nelson, M.J. (2016). *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer. ISBN 978-3-319-42714-0.

Togelius, J., Champandard, A.J., Lanzi, P.L., Mateas, M., Paiva, A., Preuss, M. and Stanley, K.O., 2013. Procedural Content Generation: Goals, Challenges and Actionable Steps. In: Lucas, S.M., Mateas, M., Preuss, M., Spronck, P., Togelius, J., eds, *Artificial and Computational Intelligence in Games*. Dagstuhl, Germany: Schloss Dagstuhl--Leibniz-Zentrum fuer Informatik, pp. 61-75.

Togelius, J., Kastbjerg, E., Schedl, D. and Yannakakis, G.N. (2011a). What is Procedural Content Generation?: Mario on the Borderline, *Proceedings of the 2Nd International Workshop on Procedural Content Generation in Games* 2011, ACM, pp. 3:6.

Togelius, J., Yannakakis G.N., Stanley, K.O., and Browne, C. (2011b) Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games*. [Online] 3, 3. 172-186. Available from: 10.1109/TCIAIG.2011.2148116 [Accessed 17.06.2017].

Yannakakis, G.N. and Togelius, J., 2011. Experience-Driven Procedural Content Generation. *IEEE Transactions on Affective Computing*, 2(3), pp. 147-161.