Jaakko Lipas

Proseduraalinen tehtävien luonti peleissä

Tietotekniikan kandidaatintutkielma

30. huhtikuuta 2020

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Jaakko Lipas

Yhteystiedot: jaollipa@student.jyu.fi

Ohjaaja: Tytti Saksa

Työn nimi: Proseduraalinen tehtävien luonti peleissä

Title in English: Procedural Quest Generation in Games

Työ: Kandidaatintutkielma

Opintosuunta: Tietotekniikka

Sivumäärä: 23+0

Tiivistelmä: Proseduraalista sisällönluontia on käytetty pelialalla 80-luvulta lähtien sekä teknisten että henkilöstön resurssirajoitusten kiertämiseen, mutta pelitehtävien proseduraalinen luonti on vielä melko vähän tutkittu ala. Tässä kirjallisuuskatsauksessa kootaan määritelmät proseduraaliselle sisällönluonnille, pelisisällölle ja pelitehtäville, tutkitaan pintapuolisesti useita proseduraalisen tehtävänluonnin metodeja, tunnistetaan alan haasteita sekä todetaan muutamia mahdollisuuksia lisätutkimukseen ja mahdollisiin sovelluksiin.

Avainsanat: Proseduraalinen sisällönluonti, pelitehtävät, suunnittelumetodit, kieliopilliset metodit, sääntöpohjaiset metodit

Abstract: Procedural content generation has been used in the game industry since the 80s to circumvent issues arising from limited technical and human resources, but the procedural generation of quests is as of yet to be widely studied. This survey collates definitions for procedural content generation, game content and quests, inspects many methods of procedural quest generation on the surface level, recognizes challenges within the area of study and suggests some opportunities for further research and possible applications.

Keywords: Procedural content generation, quests, planning methods, grammatical methods, rule-based methods

Kuviot

Sisältö

1 Johdanto

Peliala kasvaa vuosi vuodelta suuremmaksi: analyysiyhtiö Newzoon vuoden 2019 ennusteen mukaan (Wijman 2019) alan tuottokasvu on vuositasolla maailmanlaajuisesti noin 9.6%– samaten Suomessa tuotantoyhtiöiden määrän lievästä supistumisesta huolimatta peliala työllistää yhä enemmän ihmisiä (Neogames 2019). Vaikka pelialan viimeaikainen laajentuminen pienempien tuotantoyhtiöiden saralla onkin tuonut huomattavan vuotuisen pudotuksen keskimääräisiin kehityskustannuksiin, Kanadan viihdeohjelmistoyhdistyksen data (ESAC 2017) osoittaa, alustasta riippuen, yksittäisen pelin silti kustantavan keskimäärin jopa noin 12.5 miljoonaa Kanadan dollaria.

Yhtenä ratkaisuna pelien, erityisesti konsoli- ja PC-pelien (ibid.), suurten kehityskustannusten laskemiseksi on ehdotettu proseduraalista sisällönluontia; tapaa luoda pelisisältöä algoritmisesti, usein tekoälyratkaisuin (Shaker, Togelius ja Nelson 2016). Proseduraalista sisällönluontia on historiallisesti käytetty suurten sisältömäärien luomiseen peleihin, joilla on ollut nykypäivän suuriin peleihin verrattuna erittäin pienet kehitys- ja tekniset resurssit. Näistä tunnetuimpia ja aikaisimpia esimerkkejä ovat 80-luvun *Rogue* (Toy, Wichman ja Arnold 1980) ja *Elite* (Braben ja Bell 1984); hieman näitä uudempia mutta yhtä tunnettuja ovat *The Elder Scrolls II: Daggerfall* (Bethesda Softworks 1995) ja *Diablo* (Blizzard North 1997).

Tietokoneiden laskentatehon ja tallennustilan valtava kasvu sekä pelinkehitystyökalujen kehittyminen huomattavasti helppokäyttöisemmiksi viime 40 vuoden aikana ovat luoneet samanlaisen tilanteen nykypäivän itsenäisille pelinkehittäjille ja pienille kehitysyhtiöille, joita on moninkertainen määrä suuriin kehitysyhtiöihin verrattuna (ESAC 2017). Näille kehittäjille proseduraalinen sisällönluonti on ollut ja tulee myös tulevaisuudessa olemaan usein varteenotettava vaihtoehto jopa loputtomien sisältömäärien luontiin hyvin pienellä budjetilla, kuten käy ilmi esimerkiksi peleistä *Spelunky* (Yu 2008), *Minecraft* (Mojang 2009/2011) ja *Terraria* (Re-Logic 2011).

Edellämainituista esimerkkipeleistä kuitenkin vain yksi, *Daggerfall*, on käyttänyt pelitehtävien luontiin proseduraalisia metodeja; senkin päätarina ja tehtävät ovat käsintehtyjä. Vaikka proseduraalista sisällönluontia käytetään pelialalla jo yleisesti, mitä syitä on sille, ettei

pelitehtäviä luoda useammin proseduraalisesti, ja mitä metodeja on kehitetty syinä olevien haasteiden ratkaisemiseksi? Tämän tutkielman tarkoituksena on vastata näihin kysymyksiin kirjallisuuskatsauksella proseduraalisesta pelitehtävien luonnista, sivuten siihen liittyviä aiheita, kuten pelitehtävien määritelmiä sekä proseduraalista pelitarinoiden luontia.

Ennen varsinaista eri metodien esittelyä, tutkielmassa käydään läpi tarkemmin proseduraalisen sisällönluonnin, pelisisällön ja pelitehtävien määritelmiä. Erinäisten proseduraalisen pelitehtävien luonnin metodien läpikäynnin jälkeen pohditaan menetelmien haasteita, mitä nykyisen tutkimustiedon tuloksilla voitaisiin konkreettisesti tehdä sekä mitä jatkotutkimusmahdollisuuksia alalla on.

2 Pohjatieto

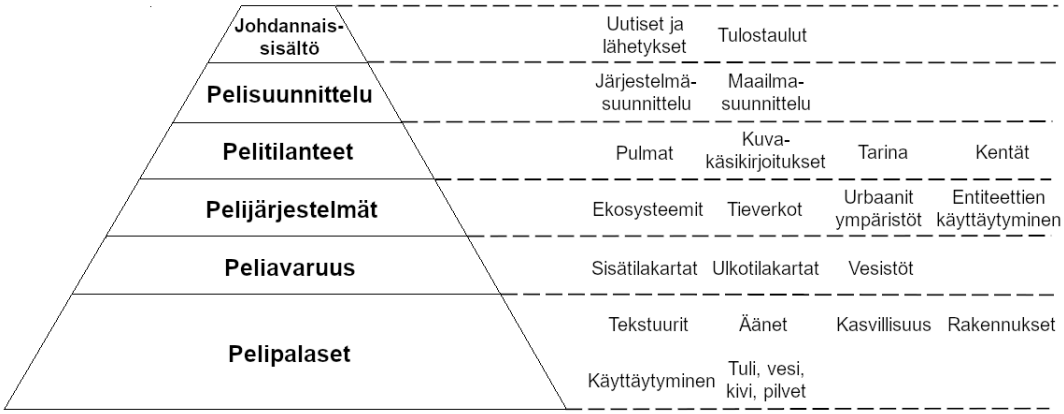
Tässä luvussa käsitellään hieman syvällisemmin määritelmät proseduraaliselle sisällönluonnille, pelisisällölle sekä pelitehtäville.

2.1 Proseduraalinen sisällönluonti

Proseduraalinen sisällönluonti peleissä (PCG-G, *Procedural Content Generation for Games*) viittaa tietokoneohjelmistoon, joka pystyy luomaan pelisisältöä algoritmisesti joko omatoimisesti tai pelaajien ja/tai suunnittelijoiden syötteen avulla (Shaker, Togelius ja Nelson 2016). Proseduraaliseen sisällönluontiin kuuluu myös vahvasti se, että jo itse algoritmit osaavat rajata tuottamansa sisällön sekä teknisesti että kulttuuriarvollisesti mielekkäisiin instansseihin (Hendrikx ym. 2013)–toisin muotoiltuna, *tekninen* ja *kulttuuriarvollinen* voitaisiin mieltää sisällön *objektiivisiksi* (esim. pystyykö luodun pelikentän ylipäänsä läpäisemään) ja *subjektiivisiksi* (esim. onko luotu pelikenttä hauska pelata) ominaisuuksiksi. Sisällön automaattinen rajaaminen täten mielekkäästi on vaatimus erityisesti peleissä, joissa sisältö muuttuu jokaisella pelikerralla, eikä ole nk. *"pre-seeded"*, jossa sisältö on proseduraalisesti luotu, mutta loppukäyttäjä pelaa aina saman, pelikehittäjän valitseman kokonaisuuden.

Togelius ym. (2011) määrittelevät rajatapausten avulla, mikä on ja ei ole proseduraalista sisällönluontia, ja päätyvät sen kautta hyvin samanlaiseen määritelmään kuin mitä Shaker, Togelius ja Nelson (2016) kirjassaan käyttävät. He kuitenkin lisäävät erityishuomiona sen, että proseduraalisena sisällönluontina ei voi pitää sitä, jos pelaaja omalla syötteellään tietoisesti luo uutta sisältöä pelin avulla, eli käyttää sisällönluontia *pelimekaanisella* tavalla.

Hyvin monia osia yksittäisen pelin sisällöstä voi luoda proseduraalisesti, mutta jotta yksittäisiä luontimetodeja voidaan tarkastella, on ensin määriteltävä ja luokiteltava se, mitä kaikkea pelisisältö oikeastaan on. Shaker, Togelius ja Nelson (2016) esittävät määritelmän, jossa pelisisältö tarkoittaa suurinta osan kaikesta, mitä peleihin sisältyy: kentät, kartat, säännöt, tekstuurit, tarinat, esineet, tehtävät, musiikki, aseet, ajoneuvot, hahmot, jne. He eivät kuitenkaan laske määritelmään itse pelimoottoria, eivätkä myöskään pelissä käytettävää ei-pelaajahahmojen tekoälyä. Sen sijaan Hendrikx ym. (2013) käsittävät pelisisällöksi pro-



Kuvio 1. Kuvaus pelisisältötyyppien hierarkiasta. Mukailtu ja suomennettu Hendrikx ym. (2013), Fig. 1.

seduraalisen sisällönluonnin kontekstissa myös pelinsisäisen tekoälyn. Tämän määritelmän mukaan proseduraalisesti luotavat pelisisältötyypit voidaan jaotella hierarkiaan, joka perustuu sisällön riippuvuuteen muusta sisällöstä (ks. Kuvio 1: ylempänä olevat sisältötyypit ja -kategoriat vaativat, että niitä alempana olevat sisällöt ovat jo olemassa).

Vaikka Hendrikx ym. (2013) eivät aseta pelitehtäviä (engl. *quests*) suoraan sisältöhierarkiaansa, he mainitsevat ne avatessaan tarinan osiota. Täten voidaan päätellä, että sikäli kun tehtävät asetettaisiin muusta sisällöstä erilleen, sijoittuisivat ne pelitilanteiden kategoriaan. Jotta pelitehtävien luontia pystyttäisiin tarkastelemaan tarkemmin, seuraavassa luvussa käsitellään, miten pelitehtäviä ja niihin liittyviä konsepteja on määritelty.

2.2 Pelitehtävät ja tehtävälliset pelit

Aarseth (2005) määrittelee pelitehtävät hyvin yksinkertaisesti ja laajasti: pelitehtävä on mikä tahansa tavoite, jonka peli antaa pelaajalle, jolla on jotain muutakin merkityksellisyyttä kuin pelkkä pisteenlaskenta tai pelaajan suorituksen arviointi. Aarsethin määritelmässä tehtävälliset pelit (engl. *quest games*) ovat mitä tahansa pelejä, joissa on määritelmän mukaisia tehtäviä. Tehtäviä on Aarsethin mukaan kolmea eri tyyppiä, joita voidaan yhdistellä: paikkariippuvaisia (pääse jonnekin), aikariippuvaisia (tee jotain jonkin aikaa) ja tavoiteriippuvaisia

(saa tai saavuta jokin asia). Tämän lisäksi Aarseth kategorisoi tehtävällisiä pelejä niiden tilallisen etenemistyylin mukaan: yksisuuntaiset käytävät, monisuuntaiset puoliavoimet keskusta ympäröivät tilat, sekä täysin avoimet maastot.

Ashmore ja Nitsche (2007) mukailevat Aarsethin määritelmää, käsitellen sitä proseduraalisesti luodun maailman kontekstissa, ja lisäten omia havaintojaan. He hahmottavat Aarsethin määritelmästä kolme ulottuvuutta: tilan, haasteet ja määränpään, sekä ehdottavat itse neljättä, miljöötä. Siinä missä Aarsethin määritelmä käsittelee tehtäviä hyvin laaja-alaisesti ja teoreettisesti, Ashmoren ja Nitschen määritelmä on lähinnä analyysi siitä, minkälaisia tehtäviä peleissä jo on.

Omassa tyyppiluokittelussaan Ashmore ja Nitsche (2007) jaottelevat tehtävät sen mukaan, miten ne liittyvät tilalliseen etenemiseen: eksplisiittiset, metarakenteen muodostavat tehtävät, jotka toimivat luodakseen laajenevan tilan ja henkilökohtaisen kasvun tunteen; moninpeleihin liittyvät, tavoitepohjaiset tehtävät, joissa kaikki pelaajat tietävät säännöt etukäteen, ja joiden läpäisy riippuu yhtä paljon toisten pelaajien voittamisesta kuin tilallisesta etenemisestä; sekä lukko-ja-avain -pulmat, jotka läpäistäkseen pelaajan on tunnistettava este (lukko) ja etsittävä jokin esine tai kyky (avain) ympäristöstään, ennen kuin eteneminen on mahdollista. Tämän luokittelun ja aiemmin käsitellyn ulottuvuuskonseptin kautta Ashmore ja Nitsche päätyvät ajatukseen siitä, että tehtävät, sikäli kun ne ovat toteutettuja peleissä, esiintyvät jonkinlaisena tilallisena siirtymänä ja etenemisenä; Aarseth (2005) ilmaisee liki saman ajatuksen siten, että tehtävät ovat elementtejä, jotka saavat pelaajan liikkumaan paikasta A paikkaan B.

Kun tarkastellaan miten proseduraalisen tehtävienluonnin metodien tutkijat määrittelevät tehtävät, monet käyttävät tarkastelupintana rooli- ja seikkailupelejä, tuoden luonnollisesti tarinan mukanaan (Doran ja Parberry 2011; Lee ja Cho 2012; Cheong ym. 2016; Chongmesuk ja Kotrajaras 2019; Soares de Lima, Feijó ja Furtado 2019). Aarseth (2005) kuitenkin kritisoi kerronnallisten elementtien hyödyllisyyttä tehtävien määrittelyssä, kutsuen niitä yleistettävän määritelmän kannalta turhiksi. Tästä lienee havaittavissa ainakin jonkinasteinen merkitysero ludologian ja tietotekniikan alojen välillä siitä, mitä kaikkea pelitehtävillä tarkoitetaan, minkä takia tehtävienluonti on useimmiten myös tarinanluontia, kuten yksittäisiä metodeja tarkastellessa tullaan havaitsemaan.

Lyhyesti kooten, proseduraalisen tehtävienluonnin tarkoituksena on siis luoda tehtävällisiä pelejä algoritmisesti, joko valmiin pohjan päälle tai yhteistyössä muiden pelisisältötyyppien luontimetodien kanssa.

3 Metodit

Tässä luvussa käsitellään ensin yleisesti proseduraalisen sisällönluonnin metodeja, tarkentaen yleisesti tehtävienluonnin metodeihin ja tarkastellen yksittäisiä tehtävienluonnin metodeja alaluokkakohtaisesti kronologisessa järjestyksessä.

Koska keskeiseltä sisällöltään eriäviä pelisisältötyyppejä on runsaasti (ks. kuvio 1), lienee itsestään selvää, ettei samoja sisällönluontimetodeja pysty käyttämään monellekaan sisältötyypille. Vaikka alan tulevaisuudenvisiona on toteuttaa sisältögeneraattoreita, jotka pystyisivät luomaan kaiken tarvittavan pelisisällön, tai jopa omat parametrinsa, ts. koko pelikehyksen sekä myös kaiken pelisisällön (Shaker, Togelius ja Nelson 2016), nämä eivät vielä nykyteknologialla ja nykytiedolla ole saavutettavissa.

Hendrikx ym. (2013) luokittelevat sisältötyyppien lisäksi myös itse luontimetodit, luoden kuusi pääluokkaa: pseudosatunnaislukugeneraattorimetodit (engl. *Pseudo-Random Number Generators*), generatiivisten kielioppien metodit (engl. *Generative Grammars*, esim. Lindenmayer-systeemit), kuvasuodatusmetodit (engl. *Image Filtering*), tilalliset algoritmit (engl. *Spatial Algorithms*, esim. fraktaalit), kompleksisten systeemien mallinnus ja simulaatio (engl. *Modeling and Simulation of Complex Systems*, esim. agenttipohjaiset simulaatiot) sekä tekoälymetodit (engl. *Artificial Intelligence*, esim. geneettiset algoritmit). Käsitellessään pelitarinoiden proseduraalista luontia, Hendrikx ym. (2013) mainitsevat metodiluokista tekoälyn ja generatiiviset kieliopit tehtävään soveltuvina, ja seuraavissa luvuissa esiteltävät yksittäiset metodit voitaisiin kaikki luokitella jompaankumpaan.

3.1 Suunnittelumetodit

Tekoälysuunnittelu (engl. *planning*) tarkoittaa algoritmeja, jotka etsivät toimintoketjuja, jotka tyydyttävät niille asetetut tavoitteet (Cheong ym. 2016). Tässä luvussa esiteltävät metodit pohjautuvat kaikki jollain tapaa suunnittelun periaatteeseen.

Yleisesti voidaan sanoa, että suunnittelumetodit pohjautuvat osajärjestykselliseen suunnitteluun (engl. *Partial-Order Planning, POP*) ja domain-malliin (Hendrikx ym. 2013; Cheong

ym. 2016). POP, toisin kuin täysjärjestyksellinen suunnittelu (engl. *Total-Order Planning*), ei vaadi tiettyä järjestystä toimintoketjun välitavoitteille, mikä on edellytys yhtäaikaisille sekä useilla eri tavoilla ratkaistaville tehtäville (Cheong ym. 2016). Domain-malli puolestaan on kokoelma pelimaailmassa olevista tosiasioista ja mahdollisista toiminnoista, mitä siinä voi tehdä; suunnittelualgoritmit käyttävät tätä kokoelmaa tietokantanaan tehdessään suunnitelmia, joiden polkuja tulkitaan pelitehtävinä (Cheong ym. 2016).

Doran ja Parberry (2011) analysoivat neljästä eri massaroolimoninpelistä (engl. *Massively Multiplayer Online Role-Playing Game, MMORPG*) yhteensä 750 eri tehtävää, ja loivat niiden pohjalta tehtäviä antavien ei-pelaajahahmojen motivaatioihin perustuvan luokittelun, havaittuaan yleisiä teemoja aineistona käyttämistään tehtävistä. Käyttäen tätä luokittelua perustana, he loivat rajoitettuja suunnitteluongelmia ratkaisevan tehtävägeneraattorin, kuitenkaan menettämättä yleistettävyyttä verrattuna ihmisten luomiin tehtäviin. Heidän analyysinsä mukaan ihmisten luomilla tehtävillä on rakenteellisia kaavoja, jotka toistuvat ennalta arvattavissa tilanteissa, ja joilla on kaikilla implisiittiset esi- ja jälkivaatimukset, luoden pohjan tehtävägeneraattorille, joka pystyy vapaasti lisäämään sivutehtäviä ja/tai lisäaskelia niin kauan kuin se ei riko näitä vaatimuksia.

Doranin ja Parberryn tehtävägeneraattori käyttää heidän luokittelustaan löytyviä kaavoja luomaan loogisesti eteneviä tehtäviä, jotka on jaoteltu motivaatioiden lisäksi strategioihin. Strategiat sisältävät sääntöjä, ja voivat sisältää itsessään atomisia toimintoja, joilla kaikilla on em. esi- ja jälkivaatimukset, jotka määrittelevät, kuinka tehtävä käytännössä etenee. Säännöt voivat sisältää lisää sääntöjä, ja lopulta päättyvät toimintoihin. Nämä ketjut kuljettavat tehtävää eteenpäin. Esimerkikkitapauksena otettakoon motivaatio ”tieto” ja sen strategia ”vakoile”, jolla on vain yksi sääntö: vakoilu. Sääntö pitää sisällään säännön ”mene paikkaan”, toiminnon ”vakoile”, säännön ”mene paikkaan” ja toiminnon ”raportoi”; luonnollisella kielellä ilmaistuna yksinkertaisin tapaus tästä esimerkistä olisi ”mene jonnekin, vakoile jotakuta, palaa takaisin ja raportoi, mitä sait tietää”. Tätä rakennetta käyttäen generaattori pystyy luomaan hyvinkin pitkiä tehtäväketjuja, sillä useamman säännön sisällä on mahdollisuus uuden sivutehtävän alkamiseen.

Lee ja Cho (2012) käyttävät aiempaa analysoitua aineistoa (Doran ja Parberry 2011) oman metodinsa pohjana, valiten lähestymistavakseen käyttää Petri net -esitystapaa (Petri ja Rei-

sig 2008) kuvatakseen tehtävien rakenteita. Muokaten Petri net:in peruselementtejä, Lee ja Cho toteuttavat yksinkertaisen Petri net -pohjaisen tehtävägeneraattorin, jonka tulokset he todistavat toimiviksi syöttämällä ne manuaalisesti Neverwinter Nights -pelin tehtäväeditoriin. Heidän mukaansa tällä lähestymistavalla on etuna synkronisaatio tehtävätapahtumien välillä, modulaarisuus sekä kyky analysoida monimutkaisempia ympäristömalleja kuin matemaattisilla malleilla.

Breault, Ouellet ja Davies (2018) loivat CONAN (*Creation of Novel Adventure Narrative*) -nimisen domain-malliin ja suunnitteluun pohjautuvan tarina- ja tehtävägeneraattorin. Heidän domain-mallinsa käyttää, samoin kuin Lee ja Cho (2012), aiemmin esiteltyä ihmisten tekemien tehtävien analyysiä (Doran ja Parberry 2011) määritelläkseen CONAN:in sisältämät mahdolliset toiminnot ja rakenteet. Aiempiin metodeihin verrattuna CONAN eroaa huomattavasti siinä, että se käyttää iteratiivista prosessia, luoden jatkuvasti uusia suunnitelmia perustuen senhetkiseen maailman tilaan, muokaten tehtävien mahdollisia haaroja reaktiivisesti pelaajan toiminnan mukaan.

3.2 Kieliopilliset ja sääntöpohjaiset metodit

Kieliopillisissa ja sääntöpohjaisissa metodeissa lähestymistapa tehtävienluontiin on formalisoida pelimekaniikka kieliopiksi ja/tai joukoksi sääntöjä, jonka avulla luodaan algoritmisesti pelitehtäviä.

Ashmore ja Nitsche (2007) loivat oman pelitehtävien määritelmänsä (ks. luku 2.2) lisäksi graafina esitettävän sääntöpohjaisen tehtävägeneraattorin, joka luo lukko-ja-avain -tyyppisiä tehtäviä. Luonti tällä metodilla tapahtuu pelimaailman laajetessa, jolloin generaattori päättää ajonaikaisesti, luodako uuden tehtävän, ja millä säännöillä. Tätä varten generaattorin on tiedettävä, mitä ”lukkoja” ja ”avaimia” maailmassa on jo valmiiksi, jotta uusien lisäysten myötä kaikki ”lukot” on edelleen mahdollista avata ja ettei ”avaimia” ole aseteltu liian lähelle toisiaan.

Dormans (2010) kuvaa tavan luoda tehtäviä graafikieliopillisesti, tehden pelielementeistä ja ohjaustoiminnoista solmuja, jotka yhdistyvät toisiinsa säännöiksi, joista muodostuu täysi tehtäväkokonaisuus. Rakenteellisesti metodi on samankaltainen kuin aiemmin mainittu

suunnittelumetodi (Doran ja Parberry 2011), mutta pohjautuu tekoälysuunnittelun sijaan generatiiviseen kielioppiin. Tämä tehtävänluontimetodi toimii yhteistyössä toisen sisällönluontimetodin kanssa, joka luo pelitilan muotokieliopin sekä luodun tehtävän avulla. Dormans esittää myös mahdollisuuden tätä metodia käyttämällä tehtävien ajonaikaiseen luomiseen, mukautuen pelaajan toimintoihin pelin sisällä, luoden jatkuvasti kehittyvän pelikokemuksen, joka on pelaajakohtaisesti uniikki.

Alexander ja Martens (2017) lähestyvät ongelmaa avoimen maailman peleissä esittämällä formalisoidun sääntömallin *Minecraft*:in pelimekaniikoista Ceptre-määrittelykielellä, joka kuvaa, miten pelitilanteet voivat kehittyä. Analysoimalla sääntöjen suorittamisesta syntyviä kuvaajia sekä autonomisessa että manuaalisessa suorituksessa, he löytävät laajennetun version pelin olemassaolevasta, käsin tehdystä tehtävärakenteesta. Täten he ehdottavat, että kyseistä metodia pystyttäisiin käyttämään luomaan tehtäviä peleihin, joissa on valmiit pelimekaniikat muttei tehtäviä.

Stocker ja Alvin (2018) esittävät tavan luoda epälineaarisia pelitehtäviä kieliopillisesti, samanhenkisesti kuin Dormans (2010), käyttäen substantiivi-verbipareja luodakseen toimintoja, jonka jälkeen toiminnot linearisoidaan riippuvuussuhteidensa perusteella. Täten järjestetyistä, lineaarisista toiminnoista luodaan hypergraafikuvaus, yksittäinen hypersärmä, joka kuvaa pelaajalta vaadittujen toimintojen joukkoa. Tähän lisätään mahdollisuuksia saavuttaa välituloksia ja lopputulos useilla eri tavoilla, jotka poissulkevat toisensa. Kieliopin luokittelu estää tilanteiden muodostumisen, joissa luodussa tehtävässä olisi tavoitteena jotain absurdia, esimerkiksi varastaa vuori. Lopputuloksena toteutettu tehtävägeneraattori pystyy luomaan hyvinkin monimutkaisia tehtäviä, vaikka sen parametrejä on rajoitettu mielivaltaisesti.

3.3 Synteesimetodit

Synteesimetodeilla tarkoitetaan tässä tutkielmassa useita aiemmin esiteltyjä metodeja yhdisteleviä, monihaaraisia tehtäviä luovia metodeja. Ne myös edustavat uusimpia alalla esitettyjä metodeja, jotka yrittävät löytää parannettavaa aiemmista metodeista, luoden uusia tapoja lähestyä proseduraalisen tehtävienluonnin ongelmia.

Soares de Lima, Feijó ja Furtado (2019) ehdottavat tehtävienluontimetodia, joka pohjautuu

geneettisiin algoritmeihin ja automatisoituun suunnitteluun, yhdistellen suunnittelun tarinakaarien ohjaamaan evoluutiohakustrategiaan. Metodin tarkoituksena on kyetä luomaan haluttuun kerronnalliseen rakenteeseen sopivia, johdonmukaisia tehtäviä. Itse tehtävägeneraattori on toteutettu kahdessa osassa: offline-tehtävägeneraattori (engl. *Offline Quest Generator*), joka on vastuussa itse tehtävien luonnista ja geneettisen algoritmin ajosta, sekä pelimanageri (engl. *Game Manager*), joka toteuttaa tehtävien suorituksen sekä päivittää pelimaailman tilaa reaaliajassa pelaajien pelatessa. Näiden taustalla on lisäksi domain-tietokanta, joka toimii tehtävägeneraattorin domain-mallina.

Chongmesuk ja Kotrajaras (2019) käyttävät samansuuntaista luontimetodia kuin Breault, Ouellet ja Davies (2018), mutta lähestyvät tehtävienluontia hieman eri näkökulmasta: siinä missä CONAN pyrki luomaan yhtäaikaisesti kaikentyyppisiä tehtäviä, Chongmesukin ja Kotrajarasin monipolkutehtävägeneraattori (engl. *Multi-Path Quest Generator, MPQ-Generator*) luo mahdollisimman monihaaraisia, annetun tyyppisiä tehtäviä, yhdistellen useampia (Doran ja Parberry 2011; Breault, Ouellet ja Davies 2018) rakenteellisia määritelmiä omaan pelitilapohjaiseen järjestelmäänsä. Tätä jatkuvan analyysin rakennetta rajaamalla tietyin säännöin *MPQ-Generator* pystyy tuottamaan aiempia metodeja joustavampia tehtäviä sekä antaa sen käyttäjälle mahdollisuuden analysoida luotuja tehtäviä.

4 Haasteet

Tässä luvussa käsitellään proseduraalisen tehtävienluonnin haasteita, esiteltyjen metodien analyysejä sekä vähän tutkittuja mutta tärkeitä ongelmia.

Proseduraalisessa sisällönluonnissa yleisiksi haasteiksi on tunnistettu metodien nopeus, luotettavuus, ohjattavuus, ilmaisukyky ja monimuotoisuus (engl. *expressivity and diversity*) sekä luovuus ja uskottavuus (Shaker, Togelius ja Nelson 2016). Jokaista proseduraalisen sisällönluonnin metodia voidaan ajatella ratkaisuna näihin ynnä muihin mahdollisiin sisällönluontiongelmiin, mutta metodeilla joudutaan yleensä tekemään kompromissejä sen välillä, mihin haasteisiin pyritään parhaiten vastaamaan (Shaker, Togelius ja Nelson 2016). Edellämainituista nopeushaaste on yleinen kaikille proseduraalisen sisällönluonnin metodeille, mutta riippuu sisältötyypin sijaan täysin siitä, miten reaaliaikaisesti sisältöä on tarkoitus luoda. Sisällönluonnissa saa kestää jopa päiviä, jos sisältöä ei ole tarkoitus luoda ajonaikaisesti, vaan ainoastaan kehittäjäpuolella–vastaavasti jatkuvalla, pelinaikaisella luonnilla on millisekuntien tason nopeusvaatimus, esimerkiksi proseduraalisesti luodussa kasvillisuudessa (Shaker, Togelius ja Nelson 2016).

Tehtävienluonnissa olennaisimmat haasteet ovat ilmaisukyky ja monimuotoisuus sekä luovuus ja uskottavuus: itseään toistavat, yksinkertaiset tehtävät aiheuttavat turhautumista ja toimintakyvyn (engl. *agency*) tunteen hälvenemistä (Johnson 2015), mutta tehtäviä ei pitäisi myöskään selkeästi pystyä tunnistamaan algoritmisesti luoduiksi. Näiden olennaisten haasteiden huomioon ottamista voisi luvun 2.1 mukaisesti kutsua kulttuuriarvollisesti merkityksellisen sisällön tuottamiseksi.

Mainituista tehtävienluonnin metodeista kolmen suorituskyky mainittiin tarkasti: Stockerin ja Alvinin epälineaarinen tehtävägeneraattori pystyy luomaan tietyillä parametrirajoitteilla jopa 612 000 tehtävää 344 sekunnissa (Stocker ja Alvin 2018), siinä missä Chongmesukin ja Kotrajarasin *MPQ-Generator* skaalautuu hyvin huonosti, suoritusajan noustessa 14 minuutilla (1400% hitaampi) vain 10% lisäyksellä tehtäväpolkumäärään analyysissä käytettyyn luotuun sisältöön verrattuna (Chongmesuk ja Kotrajaras 2019). Toisaalta, Soares de Lima, Feijó ja Furtado (2019) osoittavat, että tietyillä optimisaatioilla (monisäikeisyys, suunnitelmien vä-

limuistitus) heidän metodinsa suoritusaika 100 peliobjektille pienenee huomattavasti–27.56 sekunnista (keskihajonta 13.21 s) ilman optimisaatioita 4.31 sekuntiin (keskihajonta 3.72 s).

Joissakin käsitellyissä metodeissa soveltuvuutta on analysoitu kvantitatiivisesti: kuinka paljon erilaisia tehtäviä tai kuinka monimuotoisia tehtäviä niillä pystytään luomaan, käyttötarkoituksesta riippuen (Alexander ja Martens 2017; Stocker ja Alvin 2018; Breault, Ouellet ja Davies 2018; Chongmesuk ja Kotrajaras 2019). Sikäli kun näissä analyyseissä on käytetty verrokkina kaupallista peliä, ovat metodit onnistuneet tavoitteissaan, luoden monimuotoisempia tehtäviä; esimerkiksi Stockerin ja Alvinin metodin luomien tehtävien keskimääräinen ratkaisutapojen määrä on 25.0, kun vastaesimerkkinä käytetyn *The Elder Scrolls V: Skyrim*:in on 3.09 (Stocker ja Alvin 2018).

Kuitenkin vain yhden metodin yhteydessä (Soares de Lima, Feijó ja Furtado 2019) on kokeellisesti arvioitu pelitehtävien kannalta tärkeintä eli kvalitatiivista ulottuvuutta, toisin sanoen sitä, onko pelaajien mahdollista erottaa ihmiskehittäjän ja luontimetodin tekemiä pelitehtäviä toisistaan sokkotestissä. Soares de Lima, Feijó ja Furtado (2019) käyttivät metodiaan olemassaolevaan peliin, johon lisättiin myös ammattilaispelisuunnittelijan luomia tehtäviä, ja ryhmää opiskelijoita (N = 34) pyydettiin tunnistamaan, mitkä tehtävät olivat suunnittelijan ja mitkä proseduraalisen metodin luomia. Lopputuloksena tehtäviä ei pystytty selkeästi erottamaan toisistaan: generoidut tehtävät tunnistettiin oikein 48 kertaa 102:sta, ja ihmisen luomat tehtävät 52 kertaa 102:sta (49.02% tarkkuus). Doran ja Parberry (2011) sekä Breault, Ouellet ja Davies (2018) tunnustavat tarpeen tämäntyyppiselle analyysille, jotta proseduraalisesti luotuja tehtäviä voitaisiin pitää varteenotettavina ihmisten tekemiin tehtäviin verrattuina.

Joitakin metodeja on vaikea analysoida millään konkreettisella tavalla perustoimivuuden toteamisen lisäksi. Hyvä esimerkki tästä on *Charbitat*-pelissä käytetty generaatiometodi (Ashmore ja Nitsche 2007), jonka ajonaikaisuuden ja pelaajakohtaisen uniikkiuden vuoksi suurikaan testaajamäärä ei tuota vertailukelpoista dataa, minkä lisäksi testaajilla voi olla erilaisia käsityksiä siitä, mitä pelitehtävillä tarkoitetaan (kuten myös itse tutkijoilla, ks. luku 2.2). Ashmore ja Nitsche mainitsevatkin, että yksi lupaava tutkimusmahdollisuus olisi sellaisen evaluaatioviitekehyksen luonti, joka pystyisi yleisesti analysoimaan mitä tahansa pelitehtäviä luontimetodista riippumatta. Tutkimuksen iästä nopeasti kehittyvällä alalla huolimatta, tällaista ei ole vielä kehitetty, vaan proseduraalisten tehtävienluontimetodien tutkijat käyttä-

vät pääsääntöisesti vain omiin metodeihinsa kehitettyjä analyysejä.

Tähän mennessä mainittujen haasteiden lisäksi yleinen ongelma tehtävienluontimetodeissa on käytettävyys. Vaikka niistä monet todistettavasti toimivat tavoitteidensa mukaisesti, muutamassa tutkimuksessa jatkotutkimuksen suhteen suurin ongelma on metodin integraatio valmiiseen peliin tai peliprototyyppiin (Doran ja Parberry 2011; Lee ja Cho 2012; Stocker ja Alvin 2018), siinä missä vain Ashmore ja Nitsche (2007) sekä Soares de Lima, Feijó ja Furtado (2019) implementoivat metodinsa olemassaoleviin peleihin.

5 Yhteenveto

Tässä tutkielmassa on käyty läpi proseduraalisen sisällönluonnin, pelisisällön ja pelitehtävien määritelmät, joiden avulla on esitetty lyhyesti kirjo proseduraalisen tehtävienluonnin metodeja. Tutkielmassa on myös otettu huomioon metodien tekniset, kvantitatiiviset ja kvalitatiiviset analyysit sekä yleiset proseduraalisen sisällönluonnin haasteet.

Proseduraalista tehtävienluontia on tutkittu viime vuosina aiempaa enemmän, eikä se ole enää yhtä alisteista tarinanluonnille kuin aikaisemmin. Silti alalla on yhä todetusti suuri käytettävyyden ongelma. Useampia metodeja tulisi jatkossa integroida olemassaoleviin peleihin tai niitä varten luoda pelattavia prototyyppejä, jotta niiden selviytymistä luovuuden ja uskottavuuden haasteista voitaisiin tutkia kokeellisesti. Sikäli kun useilla metodeilla pystyttäisiin saavuttamaan samanlaisia kvalitatiivisia tuloksia kuin Soares de Lima, Feijó ja Furtado (2019) saivat omalla metodillaan, mutta myös monimutkaisemmissa peleissä ja suuremmalla otannalla, olisivat proseduraalisesti luodut tehtävät vakavasti otettava vaihtoehto henkilöstörajoitetuille kehittäjille lähes pelistä riippumatta.

Muita mahdollisuuksia lisätutkimukseen voivat tuoda edistysaskeleet muilla tietotekniikan aloilla, kuten koneoppimisessa ja tekoälyn saralla yleensä. OpenAI:n luomaa GPT-2-mallia (Radford ym. 2019) on jo käytetty kehittämään *AI Dungeon*-niminen, loputtomasti tarinasisältöä tuottava tekstiseikkailupeli (Walton 2019), joten on mahdollista, että sitä pystyttäisiin hyödyntämään proseduraalisen tehtävienluonnin metodien kehittämisessä. Torrado ym. (2019) ehdottavat neuroverkkojen hyödyntämistä suoraan esimerkiksi tehtävien luomisessa, mutta vakaviksi ongelmiksi muodostuvat neuroverkolle annettavan oppimismateriaalin vähyys ja/tai se, ettei materiaali ole julkisesti saatavilla, sekä täydellisen teknisen toimivuuden pakollisuus.

Lähteet

Aarseth, Espen. 2005. “From Hunt the Wumpus to EverQuest: Introduction to Quest Theory”. Teoksessa *Entertainment Computing - ICEC 2005,* 496–506. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-32054-8.

Alexander, Ryan, ja Chris Martens. 2017. “Deriving Quests from Open World Mechanics”. Teoksessa *Proceedings of the 12th International Conference on the Foundations of Digital Games.* FDG ’17. Hyannis, Massachusetts: Association for Computing Machinery. ISBN: 9781450353199. doi:10.1145/3102071.3102098. https://doi.org/10.1145/3102071.3102098.

Ashmore, Calvin, ja Michael Nitsche. 2007. “The Quest in a Generated World”. Teoksessa *Proceedings of the 2007 DiGRA International Conference: Situated Play.* University of Tokyo. http://www.digra.org/wp-content/uploads/digital-library/07311.20228.pdf.

Bethesda Softworks. 1995. *The Elder Scrolls II: Daggerfall.* PC.

Blizzard North. 1997. *Diablo.* PC.

Braben, David, ja Ian Bell. 1984. *Elite.* BBC Micro/Acorn Electron.

Breault, Vincent, Sebastien Ouellet ja Jim Davies. 2018. *Let CONAN tell you a story: Procedural quest generation.* arXiv: 1808.06217 [cs.AI].

Cheong, Yun-Gyung, Mark O. Riedl, Byung-Chull Bae ja Mark J. Nelson. 2016. “Procedural Content Generation in Games”. Luku 7, "Planning with applications to quests and story", toimittanut Noor Shaker, Julian Togelius ja Mark J. Nelson, 123–141. Springer Nature.

Chongmesuk, Thongtham, ja Vishnu Kotrajaras. 2019. “Multi-Paths Generation for Structural Rule Quests”. Teoksessa *2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE),* 97–102. doi:10.1109/JCSSE.2019.8864168.

Doran, Jonathon, ja Ian Parberry. 2011. “A Prototype Quest Generator Based on a Structural Analysis of Quests from Four MMORPGs”. Teoksessa *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games.* PCGames ’11. Bordeaux, France: Association for Computing Machinery. ISBN: 9781450308724. doi:10.1145/2000919.2000920. https://doi.org/10.1145/2000919.2000920.

Dormans, Joris. 2010. “Adventures in Level Design: Generating Missions and Spaces for Action Adventure Games”. Teoksessa *Proceedings of the 2010 Workshop on Procedural Content Generation in Games.* PCGames ’10. Monterey, California: Association for Computing Machinery. ISBN: 9781450300230. doi:10.1145/1814256.1814257. https://doi.org/10.1145/1814256.1814257.

Hendrikx, Mark, Sebastiaan Meijer, Joeri Van Der Velden ja Alexandru Iosup. 2013. “Procedural Content Generation for Games: A Survey”. *ACM Trans. Multimedia Comput. Commun. Appl.* (New York, NY, USA) 9 (1). ISSN: 1551-6857. doi:10.1145/2422956.2422957.

Hiltunen, KooPee, Suvi Latva ja J-P Kaleva. 2019. *The Game Industry Of Finland Report 2018.* http://www.neogames.fi/wp-content/uploads/2019/04/FGIR-2018-Report.pdf.

Johnson, Daniel. 2015. “Animated Frustration or the Ambivalence of Player Agency”. *Games and Culture* 10 (6): 593–612. doi:10.1177/1555412014567229. https://doi.org/10.1177/1555412014567229.

Lee, Young-Seol, ja Sung-Bae Cho. 2012. “Dynamic Quest Plot Generation Using Petri Net Planning”. Teoksessa *Proceedings of the Workshop at SIGGRAPH Asia,* 47–52. WASA ’12. Singapore, Singapore: Association for Computing Machinery. ISBN: 9781450318358. doi:10.1145/2425296.2425304. https://doi.org/10.1145/2425296.2425304.

Re-Logic. 2011. *Terraria.* PC.

Mojang. 2009/2011. *Minecraft.* PC.

Petri, Carl Adam, ja Wolfgang Reisig. 2008. “Petri net”. Revision #91647, *Scholarpedia* 3 (4): 6477. doi:10.4249/scholarpedia.6477.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei ja Ilya Sutskever. 2019. *Language Models are Unsupervised Multitask Learners.* San Francisco, CA, USA. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

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

Soares de Lima, Edirlei, Bruno Feijó ja Antonio L. Furtado. 2019. “Procedural Generation of Quests for Games Using Genetic Algorithms and Automated Planning”. Teoksessa *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames),* 144–153. doi:10.1109/SBGames.2019.00028.

Stocker, Alex, ja Chris Alvin. 2018. “Non-Linear Quest Generation”. https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS18/paper/view/17606/16885.

The Entertainment Software Association of Canada. 2017. *Essential Facts 2017.* http://theesa.ca/wp-content/uploads/2017/10/ESAC2017_Booklet_13_Digital.pdf.

Togelius, Julian, Emil Kastbjerg, David Schedl ja Georgios N. Yannakakis. 2011. “What is Procedural Content Generation? Mario on the Borderline”. Teoksessa *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games.* PCGames ’11. Bordeaux, France: Association for Computing Machinery. ISBN: 9781450308724. doi:10.1145/2000919.2000922. https://doi.org/10.1145/2000919.2000922.

Torrado, Ruben Rodriguez, Ahmed Khalifa, Michael Cerny Green, Niels Justesen, Sebastian Risi ja Julian Togelius. 2019. *Bootstrapping Conditional GANs for Video Game Level Generation.* arXiv: 1910.01603 [cs.NE].

Toy, Michael, Glenn Wichman ja Ken Arnold. 1980. *Rogue.* UNIX.

Walton, Nick. 2019. *AI Dungeon 2.* PC/iOS/Android. https://aidungeon.io/.

Wijman, Tom. 2019. *The Global Games Market Will Generate $152.1 Billion in 2019 as the U.S. Overtakes China as the Biggest Market.* https://newzoo.com/insights/articles/the-global-games-market-will-generate-152-1-billion-in-2019-as-the-u-s-overtakes-china-as-the-biggest-market/.

Yu, Derek. 2008. *Spelunky.* PC.