Tampereen yliopisto

Johannes Hovinen

LAADUKKAIDEN VIDEOPELIKENTTIEN TUOTTAMINEN GENERATIIVISILLA KILPAILEVILLA NEUROVERKOILLA

Kandidaatintyö
Informaatioteknologian ja viestinnän tiedekunta
Toukokuu 2022

TIIVISTELMÄ

Johannes Hovinen: Laadukkaiden videopelikenttien tuottaminen generatiivisilla kilpailevilla neuroverkoilla
Kandidaatintyö
Tampereen yliopisto
Tietotekniikan tutkinto-ohjelma
Toukokuu 2022

Videopelien tuotannossa videopelikenttien luominen on yksi resurssi-intensiivimimmistä prosesseista. Ajan ja rahan säästämiseksi videopelikenttiä voidaan tuottaa automaattisesti. Historiallisesti algoritmeilla on tuotettu videopelikenttiä esimerkiksi roguemaisiin videopeleihin, joissa peliin lisätään uudelleenpelattavuutta tuottamalla uusia kenttiä jokaisella pelauskerralla.

Videopelikenttä sisältää pelaajahahmon, tavoitteet ja esteet. Pelaajahahmon täytyy päästä tavoitteeseen ilman epäonnistumista ja esteet pyrkivät luomaan pelaajalle haastetilanteita.

Videopelikentän tuottaminen automaattisesti vähentää suunnittelijan vaikutusta lopputulokseen. Jotta voidaan tuottaa suuri määrä erilaisia videopelikenttiä, täytyy automaattisessa ratkaisussa olla tarpeeksi satunnaisuutta. Satunnaisuus voi myös vaikuttaa pelikokemukseen tai pelattavuuteen negatiivisesti. Esimerkiksi liian vaikea yhdistelmä esteitä voi aiheuttaa pelaajalle vaikeuksia edetä.

Automaattista videopelikentän tuottamista on myös alettu tutkia koneoppimisen näkökulmasta. Sisällöntuotanto koneoppivin keinoin pyrkii luomaan uutta jo olemassa olevasta datasta. Koneoppivien metodien soveltaminen videopelikenttien tuottamiseen ei kuitenkaan ole suoraviivaista, sillä videopelikenttien pelattavuuden rajoitteet aiheuttavat ongelmia. Tässä työssä tarkastellaan miten pelattavuuteen, ulkonäköön ja vaikeusasteeseen voidaan vaikuttaa generatiivisten kilpailevien verkkojen yhteydessä.

Generatiiviset kilpailevat verkot ovat koneoppimisen metodi, joka perustuu kahden neuroverkon kilpailuun. Nämä verkot pystyvät onnistuneesti tuottamaan esimerkiksi kuvia, kuitenkin videopelikenttien pelattavuuden rajoitteiden oppiminen aiheuttaa ongelmia. Näiden pelattavuuden rajoitteiden toteutumiseksi neuroverkkojen lisäksi voidaan käyttää esimerkiksi geneettistä optimointia tai videopelikentän korjausta. Myös neuroverkon arkkitehtuurilla ja aloitusdatan manipulaatiolla voidaan vaikuttaa positiivisesti lopputulokseen.

Tutkielmassa todettiin, että tarpeeksi sivistyneellä ratkaisulla generatiivisilla kilpailevilla verkoilla voidaan tuottaa yksinkertaisia videopelikenttiä, jotka näyttävät vakuuttavilta ja täyttävät pelattavuuteen liittyvät rajoitteet. Kuitenkin yleisistä pelien välisistä toteutuksista ollaan vielä kaukana ja suurin osa tutkimuksesta rajoittuu kaksiulotteisiin diskreetteihin videopeleihin.

Avainsanat: proseduraalinen sisällöntuotanto, koneoppiminen, videopelikentät, generatiiviset kilpailevat verkot

SISÄLLYSLUETTELO

LYHENTEET

GAN	Generatiivinen kilpaileva verkko (engl. generative adversarial network)
PCG	Proseduraalinen sisällöntuotanto
PCGML	Proseduraalinen sisällöntuotanto koneoppivin keinoin (engl. procedural content generation via machine learning)

1. JOHDANTO

Manuaalinen videopelikentän tuottaminen on yksi videopelikehityksen resursseja ja aikaa vaativimmista osista. Pääsyy tälle on se että videopelikenttien pitäisi olla vähimmäislaatuisia tasapainon, vaikeusasteen ja ulkonäön suhteen (Davoodi ja muut, 2020). Proseduraalinen sisällöntuotanto *(Procedural Content Generation, PCG)* onkin tästä syystä jatkuvasti näkyvämpi osa-alue videopelien tuotannossa ja sitä on käytetty onnistuneesti suosituissa videopeleissä kuten Terraria (Re-Logic, 2011), Minecraft (Mojang, 2011) ja Spelunky (Mossmouth, 2008). PCG:llä tarkoitetaan sisällön tuottamista algoritmisesti ja se on usein automaattista. PCG:n päätavoitteisiin kuuluu: uudelleenpelattavuuden lisääminen, tuottamiskustannuksien vähentäminen ja joskus muistin säästäminen. PCG:n algoritmisestien ratkaisujen toteuttaminen ei kuitenkaan ole triviaalia ja voi pelin ja siihen tuotetun sisällön monimutkaisuuden mukaan vaatia paljon resursseja. Tästä syystä on alettu tutkia koneoppivia ratkaisuja sisällöntuotannon ongelmiin.

Koneoppimisen suosion kasvaessa sitä on alettu hyödyntää sisällöntuotannossa. PCG:n rinnalle on kehittynyt uusi tutkimusala: proseduraalinen sisällöntuottaminen koneoppivin keinoin *(Procedural Content Generation via Machine Learning, PCGML)*, jolla pyritään automatisoimaan videopelikenttien tuottamisen prosessia käyttäen koneoppivia malleja. Koneoppimisen avulla pystytään tunnistamaan jo olemassa olevasta sisällöstä piirteitä, joiden avulla voidaan luoda uutta sisältöä. Päämotivaatioina koneoppivissa ratkaisuissa videopelien yhteydessä ovat ajan ja rahan säästö, mahdollisuus luoda loputtomasti sisältöä ja mahdollisesti sopeutuvien pelien luonti (Liu ja muut, 2020). Koneoppivien metodien soveltaminen videopelikenttien tuottamiseen on kuitenkin vasta alkuvaiheessa ja monimutkaisten kenttien tuottaminen on erittäin vaikeaa.

Yksi vakuuttavimmista PCGML-metodeista videopelikenttien luonnin yhteydessä ovat generatiiviset kilpailevat verkot. Generatiivisten kilpailevien verkkojen toiminta perustuu kahden eri neuroverkon kilpailuun. Generatiivisilla kilpailevilla verkoilla on saatu aikaan vakuuttavia videopelikenttiä artikkeleissa Capps ja Schrum (2021),Torrado ja muut (2019) ja Zhang ja muut (2020).

Generatiivisia kilpailevia verkkoja on käytetty onnistuneesti esimerkiksi kuvien tuottamiseen, mutta videopelikenttien tuottamien sisältää paljon sääntöjä, joiden mallintaminen koneoppimisen avulla on haastavaa. Tuotetun sisällön pieni virhe ei ole välttämättä on-

gelma kuvan yhteydessä, mutta videopelitasossa mahdoton hyppy tai läpäisemätön sokkelo tekee tasosta mahdottoman läpäistä (Risi ja Togelius, 2019). Koneoppivien ratkaisujen heikkoudet on huomattu olevan kenttien pelattavuus, ulkonäkö ja monimuotoisuus. Neuroverkot eivät siis opi tarpeeksi hyvin videopelikenttien piirteitä ilman tiettyjä koneoppimistekniikoita, apufunktioita tai datan manipulointia.

Tämän tutkielman tavoitteena on selvittää, miten GAN-sovelluksissa nämä tärkeät ulkonäköön, pelattavuuteen ja vaikeusasteeseen liittyvät ongelmat on ratkaistu.

Tutkielmassa tarkastellaan ensin luvussa 2, mikä on videopelitason rakenne ja suunnittelumenetelmä yleisesti. Seuraavassa luvussa 3 tarkastellaan, miten sisällöntuottamista hyödynnetään videopelikenttien yhteydessä ja miten PCG ja PCGML eroavat toisistaan. Seuraavaksi luvussa 4 esitetään, miten generatiiviset kilpailevat verkot toimivat ja miten GAN:ia voidaan käyttää videopelikentän tuottamiseen. Luvussa 5 eritellään GAN:ien heikkoudet videopelikenttien tuottamisen yhteydessä ja miten näitä heikkouksia on pyritty ratkaisemaan.

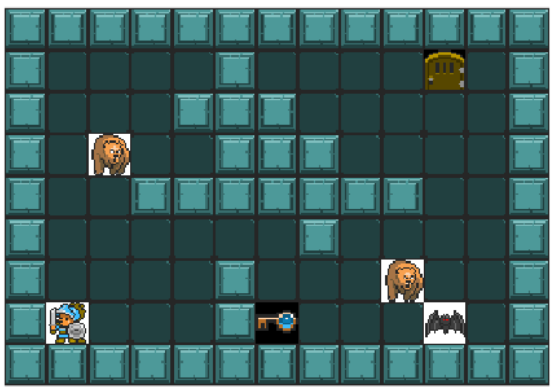
2. VIDEOPELIKENTTIEN SUUNNITTELU

Pelikentän onnistunut suunnittelu on esivaatimus hyvälle videopelille. Byrne kuvaa pelikenttiä ’pelaamisen säiliöinä’ , jotka tarjoavat pelaajalle interaktiivisen tilan ja mahdollisuuden tutkia pelimaailmaa sen sääntöjen puitteissa (Byrne, 2005). Tämän takia pelikentän suunnittelu on monimutkainen tehtävä, jonka onnistumiseen vaaditaan syvällistä ymmärrystä kaikista pelin komponenteista ja niiden suhteista (Smith ja muut, 2008). Seuraavaksi esitellään videopelikenttien rakennetta ja suunnitteluperiaatteita. Koska suurin osa tällä hetkellä PCGML:llä käsitellyistä videopeleistä on kaksiulotteisia diskreettejä tiilipohjaisia (engl. tile-based) videopelejä, kuten Super Mario Bros, The Legend of Zelda ja Mega Man, tarkastelemme vain ja ainoastaan niitä.

Komponenttierittely

Tämän kappaleen tavoitteena on selittää, mistä komponenteista pelikenttä koostuu ja mitkä ovat niiden suhteet. Komponentteja havainnollistetaan General Video Game Artificial Intelligence (GVG-AI) ympäristön Zelda-videopelillä, joka on yksinkertaistettu versio Nintendon julkaisemasta The Legend of Zelda -videopelistä. Zelda-pelin esimerkkikenttä on esitetty kuvassa 2.1 ja pelin sisältämät komponentit kuvassa 2.2.

Bjork ja Holopainen (2005) esittävät videopelien suunnitteluperiaatteet kirjassaan "Patterns in game design". *Pelikenttä* on osa videopeliä, jossa pelaajan toiminta tapahtuu, kunnes pelaaja on päässyt kentän tavoitteeseen. *Tavoite* on pelikentän läpäisyyn vaadittava edellytys, joka voi olla esimerkiksi keino päästä seuraavaan kenttään, päävastus tai



Kuva 2.1. *Esimerkki Zelda-pelin kentästä (Zhang ja muut, 2020).*

Kuva 2.2. *Zelda-pelin eri komponentit (Zhang ja muut, 2020)*

jokin esine. Tavoitteet voidaan jakaa osatavoitteisiin, jotka muodostavat *tavoitehierarkian*.

Edellä mainitun Zelda-pelin tavoitehierarkian muodostavat avain ja ovi. Avaimen saaminen on alitavoite ja sillä oven avaaminen päätavoite.

Pelikenttä sisältää *esteitä* alkupisteen ja loppupisteen eli päätavoitteen välillä, jotka tekevät pelikentästä pelaajalle haastavan. Esteisiin luokitellaan *viholliset, ansat ja pulmat*. Vihollisten määrällä ja läheisyydellä toisiinsa on yleensä suuri vaikutus pelaajan kokemaan vaikeusasteeseen. Ansoilla pyritään aiheuttamaan pelaajalle vaaratilanteita, joista pelaaja joko suoriutuu tai ei. Yleensä ansaan jäämisestä seuraa elämä- tai elämäpisterangaistus.

Zelda-pelissä ei ole varsinaisia ansoja, mutta esteiksi voidaan luokitella viholliset ja seinät.

Komponenttierittely vaihtelee eri peligenrejen välillä, mutta yleiset komponentit kuten edellä mainitut ovat useimmissa videopeleissä.

Vaikeusaste

Haaste on videopelikentän suunnittelun tärkein osa. Videopelikentän vaikeusaste täytyy pysyä loogisena, jotta pelaaja ei turhaudu peliin (Byrne, 2005). Pelaajan *taitotaso* on suoraan sidonnainen pelaajan kokemaan suhteelliseen *vaikeusasteeseen*. Pelikenttää suunnitellessa on ehdottoman tärkeää ottaa huomioon pelaajan mahdollinen taitotaso ja kentän vaikeusaste. Pelikentän ollessa liian vaikea tai liian helppo kärsii pelaajan pelikokemus.

Sujuvuus

Pelikentän sujuvuudella (engl. flow) tarkoitetaan pelikentän kykyä pitää pelaajan mielenkiinto. Pelaaja voi kokea katkoksen, jos peli yhtäkkiä muuttuu hauskasta pitkäveteiseksi, tai pelaaja ei tiedä, miten kentässä päästään eteenpäin.

3. VIDEOPELIKENTTIEN GENEROINTI

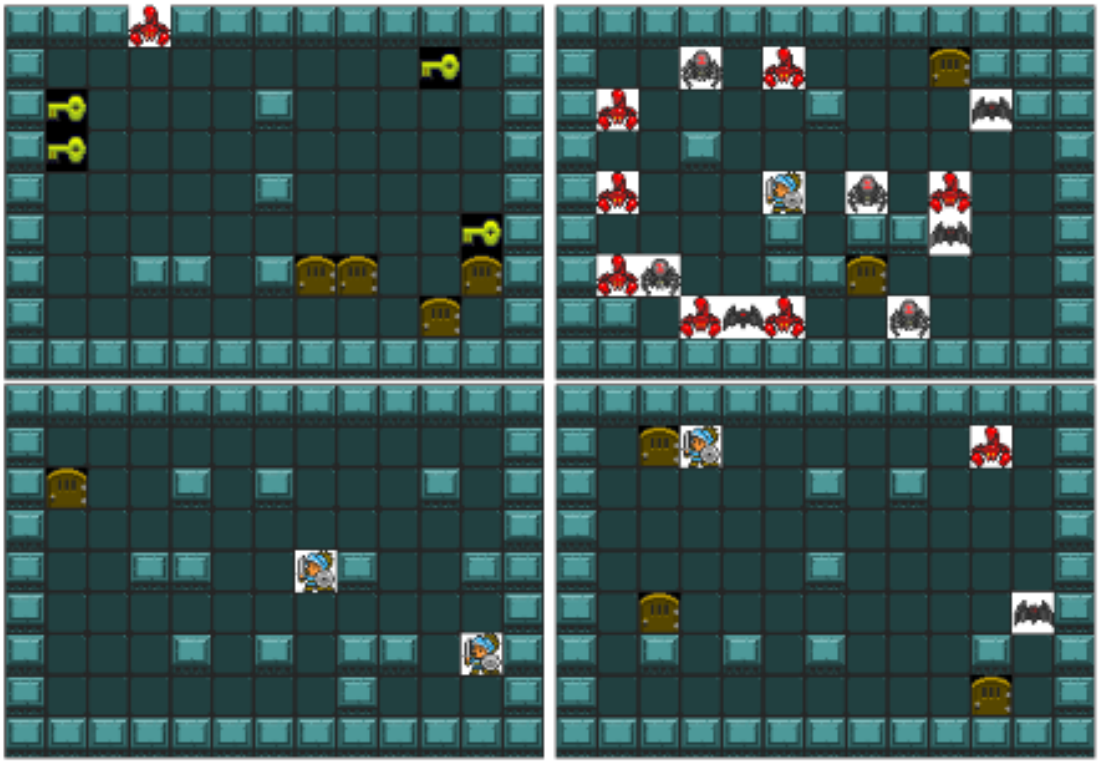
Tässä luvussa esitetään, miten sisällöntuotantoa hyödynnetään pelikenttien generoinnissa ja miten suunnittelijan vaikutus vähenee automaattisissa toteutuksissa.

3.1 Proseduraalinen sisällöntuottaminen

Proseduraalisella sisällöntuottamisella tarkoitetaan sisällön tuottamista ilman tai rajoitetulla käyttäjävaikutteella. Toisin sanoen PCG:llä tarkoitetaan tietokoneohjelmaa, joka pystyy tuottamaan pelisisältöä itse tai suunnittelijan kanssa. Peleihin on tuotettu PCG:n avulla muun muassa kenttiä, karttoja, pelisääntöjä, tekstuureita, tarinoita, esineitä, tehtäviä ja musiikkia (Shaker, 2016). PCG-tekniikoiden avulla tuotettu sisältö harvoin vastaa käsin harkiten suunniteltua ja rakennettua sisältöä, mutta tarpeeksi hienostuneella algoritmisellakin ratkaisulla voidaan saada aikaan uskottavaa sisältöä. Esimerkiksi roguemainen (engl. Roguelike) peligenre hyödyntää automaattisesti luotujen kenttien uudelleenpelattavuutta, missä kompromissi voi olla yksittäisten kenttien laatu.

Kuvassa 3.1 näkyy PCG:n avulla generoitu pelikenttä roguemaisessa videopelissä Spelunky 2 (Mossmouth, 2020). Pelin alkaessa generoidaan aina uusi uniikki pelikenttä algoritmisesti.

Kuva 3.1. *Proseduraalisesti luotu pelikenttä Spelunky 2 (Mossmouth, 2020) videopelissä.*

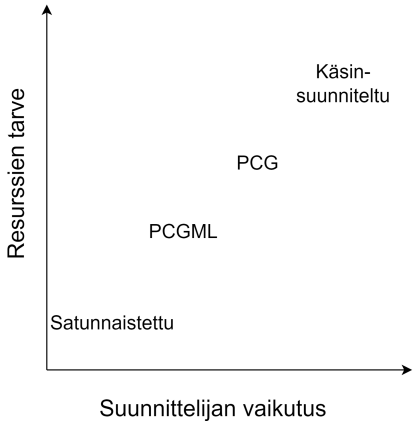


Kuva 3.2. *Neuroverkon tuottamia mahdottomia Zelda-pelikenttiä (Torrado ja muut, 2019).*

3.2 Proseduraalinen sisällöntuottaminen koneoppivin keinoin

Proseduraalinen sisällöntuottaminen koneoppivin keinoin tarkoittaa sisällöntuottamista käyttäen jo olemassa olevaa dataa, josta koneoppiva algoritmi pyrkii oppimaan piirteitä (Summerville ja muut, 2018). Koneoppivin keinoin pelikentän luonti ei ole yhtä suoraviivaista, kuin pelkästään PCG:n avulla. Suunnittelijan vaikutus pelikentän ominaisuuksiin vähenee ja verkko lisää paljon ei-haluttuja piirteitä, kuten liiallista satunnaisuutta tai mahdottomia tilanteita. Kuten artikkelissa ”Deep learning for procedural content generation” (2020), Liu ja muut summaavat: ”sopeutumistekniikoiden huolellinen harkinta ja hienostunut suunnittelu ovat edellytyksiä syväoppimismenetelmien soveltamiselle pelisisällön tuotantoon.” Sopeutumistekniikoilla tarkoitetaan niitä metodeja, joilla rajataan koneoppivan järjestelmän tuloksia eri toiminnallisuuden, kuten esimerkiksi videopelikentän pelattavuuden perusteella.

Kuvassa 3.2 on esitetty viallisia GAN:in tuottamia pelikenttiä. Zelda-pelissä pelaajan täytyy avata ovi avaimella kentän läpäisemiseksi. Alavasemmasta pelikentästä voidaan huomata, että se ei toteuta pelattavuuden rajoitteita, koska se ei sisällä ovea. Koneoppiva ratkaisu ei siis pysty mallintamaan pelattavuuden rajoitteita tarpeeksi hyvin. Yläoikeassa kentässä ilmenee avaimen puuttumisen lisäksi liian paljon vihollisia. Vihollisten suuri määrä tekee kentästä liian vaikean, joten se on lähes mahdoton läpäistä.



Kuva 3.3. *Pelikentän suunnittelumenetelmien resurssien ja suunnittelijan vaikutuksen vertailu*

3.3 Suunnittelijan vaikutus

Kun puhutaan PCG- ja PCGML-videopelikentäntuottamisen menetelmistä, videopelikenttäsuunnittelun vastuu siirtyy osaksi kooditasolle ja itse suunnittelijan vaikutus muuttuu osittain epäsuoraksi. Neuroverkot ovat yleensä ”black box” -malleja eli niiden sisäistä toimintaa on vaikea tai mahdoton ymmärtää, joten toteutus muuttuu vielä abstraktimmaksi. Kuitenkin PCGML toteutuksien yhteydessä voidaan vaikuttaa tuloksiin aloitusdatan, verkon ulostulojen muokkaamisella ja verkon arkkitehtuurin kautta. Kuvassa 3.3 näkyy, miten suunnittelijan vaikutus pelikenttään muuttuu menetelmien yhteydessä.

1. Käsintehty: Lähes jokainen pelikentän elementti on käsin suunniteltu ja sijoitettu, niin että pelaajan pelikokemus maksimoidaan. Kaikista resurssi-intensiivisin keino, mutta suunnittelijalla on eniten vaikutusta.
2. PCG: Pelikenttä luodaan algoritmisesti, usein tavoitteena on luoda satunnaisuutta ja sen kautta uudelleenpelattavuutta. Suunnittelija pystyy vaikuttamaan algoritmisella tasolla kentän rakenteeseen. Suunnittelijan vaikutus on epäsuorempaa kuin käsintehdyissä videopelikentissä ja hyvän algoritmin toteutus vie resursseja.
3. PCGML: Tuloksiin voidaan vaikuttaa epäsuorasti harjoitusdatan, ulostulon muokkaamisen ja arkkitehtuurillisten valintojen kautta. Resurssiystävällinen lähestymistapa.
4. Täysin satunnaistettu: Suunnittelija ei pysty vaikuttamaan kentän rakenteeseen.

Suunnittelijan vaikutusta voidaan lisätä PCGML-toteutuksiin esimerkiksi sopeutumistekniikoilla. Sopeutumistekniikoita kuten geneettistä optimointia ja videopelikentän korjausta tarkastellaan luvussa 5.

4. GENERATIIVISET KILPAILEVAT VERKOT

Syväoppiminen on koneoppimisen osa-alue, jossa hyödynnetään syviä neuroverkkoja. Syväoppimisessa tavoitteena on oppia piirteitä annetusta datasta ja sen perusteella joko luokitella tai luoda uusia näytteitä. Generatiiviset kilpailevat verkot ovat yksi syväoppivista malleista, joiden tavoitteena on luoda uusia näytteitä jo olemassa olevasta datasta. Generatiivisten kilpailevien verkkojen kehys esiteltiin ensimmäisen kerran artikkelissa Goodfellow ja muut (2014).

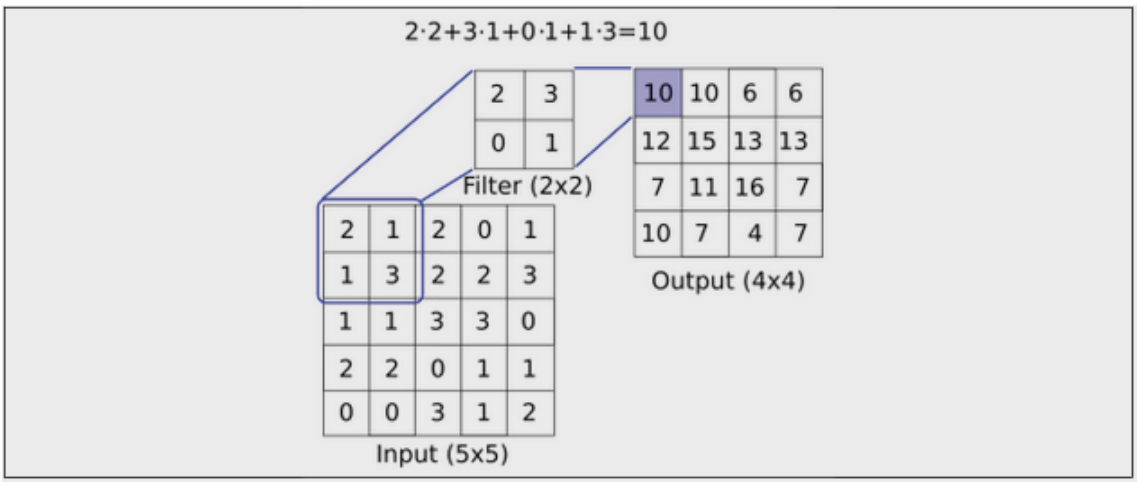
Tässä luvussa esitetään generatiivisten kilpailevien verkkojen teoria ja sen jälkeen esimerkin kautta, miten GAN-toteutusta voidaan käyttää videopelikenttien tuottamiseen.

4.1 Generatiivinen kilpaileva verkko

Generatiiviset kilpailevat verkot ovat esimerkki koneoppimisen teknologiasta, joka pohjautuu vapaasti ohjautuvaan luokitteluun. Tavoitteena vapaasti ohjautuvassa luokittelussa on tunnistaa annetusta datasta piirteitä (engl. Patterns), jotka ovat jollain tasolla olennaisia osia datassa. Ohjatussa tulkinnassa data on luokiteltu, kun taas vapaasti ohjautuvassa ei.

Generatiiviset kilpailevat neuroverkot perustuvat kahden erillisen neuroverkon kilpailuun. Toinen verkko, nimeltään generaattori, on vastuussa uuden materiaalin tuotannosta, kun taas toinen verkko, nimeltään diskriminaattori (engl. Discriminator), on vastuussa tuotetun materiaalin arvostelusta. Generaattoria voidaan mallintaa funktiona $p_{model}(x)$, jossa x on harjoitusesimerkki ei tunnetusta jakaumasta $p_{data}(x)$.

Diskriminaattori tarkastelee näytteitä x ja antaa arvion siitä onko näyte aito vai ei. Verkkojen kilpailu perustuu siihen, että generaattori oppii tuottamaan mahdollisimman aitoja epäaitoja näytteitä, jotka diskriminaattori luokittelisi aidoiksi. Diskriminaattorin tavoitteena on oppia tunnistamaan aidot ja epäaidot näytteet mahdollisimman tarkasti. Verkkojen toimintaa voidaan kuvata poliisi ja väärentäjä skenaariona, jossa diskriminaattori on poliisi ja generaattori väärentäjä. Skenaariossa poliisi pyrkii tunnistamaan väärentäjän tuottamat väärennökset ja puolestaan väärentäjä pyrkii tekemään vakuuttavampia väärennöksiä. Lopulta väärentäjä pystyy tekemään niin vakuuttavia väärennöksiä että poliisi ei pysty tunnistamaan eroja oikean ja väärennetyn välillä. Kun tähän pisteeseen on päästy, voi-



Kuva 4.1. *Esimerkki konvoluutio-operaatiosta matriisin yhteydessä (Audevart ja muut, 2021).*

daan diskriminaattori hylätä ja generaattori pystyy tuottamaan vakuuttavia näytteitä.

4.2 Konvoluutio

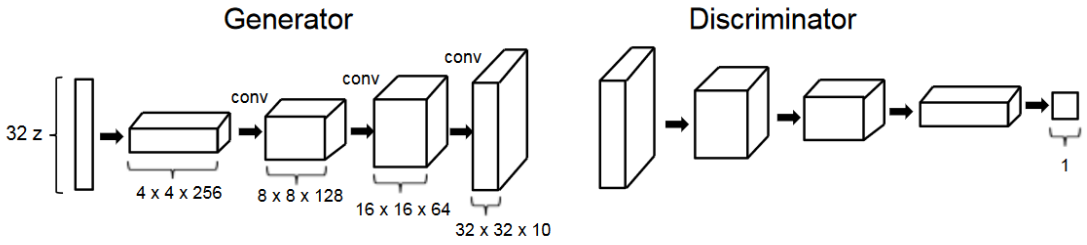
Konvoluutiota käytetään syvien neuroverkkojen yhteydessä, jotta ne oppisivat annetusta datasta matalan tason piirteitä. Yleisesti konvoluutio on funktio, jota käytetään toisen funktion ulostuloon. Kuvien ja muiden kaksiulotteisten matriisimaisten datatyyppien kanssa konvoluutio on esimerkiksi matriisien tulo, jossa konvoluutiomatriisilla suodatetaan alkuperäisestä matriisista haluttuja piirteitä (Audevart ja muut, 2021). Luonnollisesti 2-ulotteiset videopelikentät ovat matriisimaisia datatyyppejä ja niiden piirteiden oppimisen edesauttamiseksi voidaan käyttää konvoluutiota.

Konvoluutioneuroverkon yhteydessä käytetään yleensä ReLU-aktivaatio- ja aggregointikerroksia. Aggregointikerroksilla pyritään yksinkertaistamaan dataa, jotta säästetään laskentakuormaa ja muistia. Aggregointikerros voi olla esimerkiksi maxpool-kerros, jossa aina voimakkain ominaisuus säilyy seuraavaan kerrokseen. Muitakin aggregointifunktioita käytetään kuten keskiarvoa. Epälineaarisuuksia voidaan lisätä neuroverkkoon käyttämällä ReLU-aktivaatiokerroksia. Epälineaarisen datan oppiminen vaatii epälineaarisuutta verkon arkkitehtuurissa (Audevart ja muut, 2021).

Konvoluutiolla voidaan videopelikenttien yhteydessä edesauttaa pitkän etäisyyden piirteiden oppimista, kuten pelikentän sisältämien tavoitteiden suhteita.

4.3 Wasserstein-GAN

Generatiivisten kilpailevien verkkojen opettaminen on usein herkkäluontoista ja epävakaata, jonka takia Arjovsky ja muut (2017) esittivät Wasserstein-harjoitusalgoritmin.



Kuva 4.2. *Konvolutionaalisen generatiivisen kilpailevan verkon arkkitektuuri, jota käytettiin Super Mario Bros -kenttien tuottamiseen (Volz, Schrum ja muut, 2018)*

Wasserstein-GAN pyrkii myös estämään tilan romahtamisen (engl. mode collapse) harjoitusvaiheessa. Tila romahtaa, kun generaattori tuottaa vain yhtä tai muutamaa laadukasta näytettä jatkuvasti. Tilanne syntyy, kun generaattori luo erityisen uskottavan näytteen, jota diskriminaattori ei pysty tunnistamaan vääräksi. Generaattori jatkaa samojen näytteiden tuottamista ja diskriminaattori ei pysty jatkamaan. Tilan romahtamisen tapahduttua generaattori ei pysty tuottamaan monimuotoisia näytteitä ja GAN:in harjoitus epäonnistuu.

Toinen Wassersteinin hyöty on sen tappiofunktio. Maansiirto-etäisyys (engl. earth mover's distance) on aina jatkuva ja derivoitavissa, mikä tarkoittaa sitä, että diskriminaattori voidaan aina harjoittaa optimiin saakka. Wasserstein myös auttaa käyttäjää näkemään miten harjoitus edistyy. Wassersteinin tappio on suoraan sidonnainen verkon suppenemiseen ja sen tuottamien näytteiden laatuun. Standardissa GAN:issa tätä ominaisuutta ei ole.

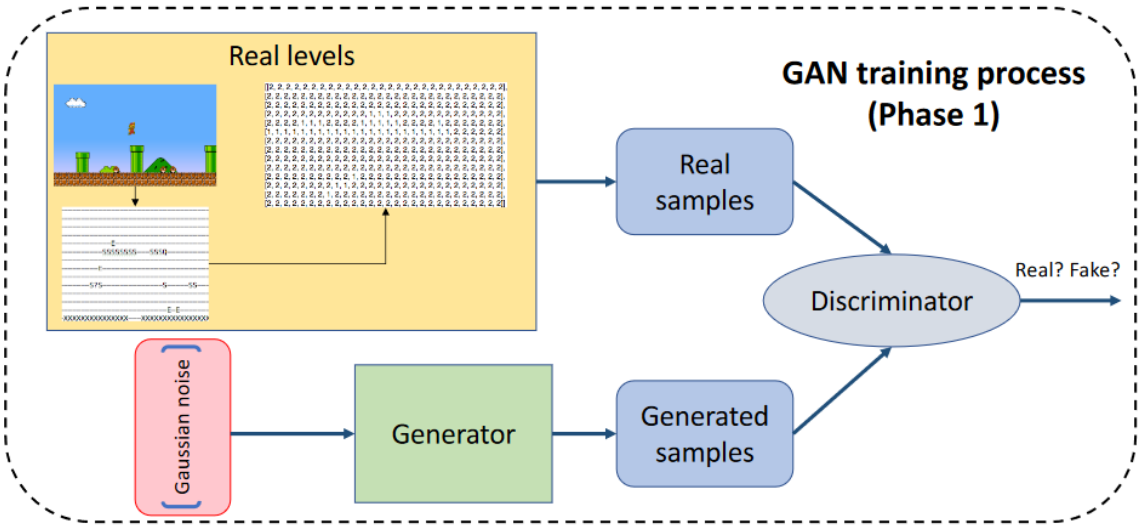
Wasserstein-GAN:ia on käytetty verkon harjoittelun edistämiseen töissä Volz, Schrum ja muut (2018), Torrado ja muut (2019), Capps ja Schrum (2021) ja Zhang ja muut (2020).

4.4 Generatiivisen kilpailevan verkon prosessi

Volz, Schrum ja muut (2018) tuottavat työssään pelikenttiä Super Mario Bros -videopeliin GAN:illa. Työssä käytetyn verkon sisäinen arkkitehtuuri on esitetty kuvassa 4.2 ja verkon harjoitusvaiheen rakenne kuvassa 4.3.

Generaattorin sisällä kohinavektorista luodaan kolmen konvolutionaalisen kerroksen jälkeen one-hot-enkoodattuja pelikenttiä. One-hot-enkoodauksessa, jokainen pelilaatan variaatio saa kokonaislukuarvon sijaan binäärivektorin. Pelikentät syötetään diskriminaattorille, joka toistaa konvoluutio-operaatiot vastakkaisessa järjestyksessä ja viimeinen ulostulo on 0 tai 1 eli epäaito tai aito.

Generaattori tuottaa näytteitä satunnaistetusta Gaussin-kohinavektorista. Generaattorin tuottamia ja harjoitusdatasta otettuja näytteitä annetaan diskriminaattorille, jotka se luokittelee luvussa 3.1 esitetyllä tavalla. Kun diskriminaattori ei enää pysty erottamaan aitoja epäaidoista se voidaan hylätä ja jäljelle jää generaattori, joka pystyy tuottamaan videope-



Kuva 4.3. *Mario pelikenttien luontiin käytetty GAN-arkkitehtuuri (Volz, Schrum ja muut, 2018)*

likenttiä.

Generaattorin sisääntulon piilomuuttujavektori (engl. latent variable vector) arvojen muuttaminen saa sen tuottamaan erilaisia pelikenttiä. Tätä yhteyttä voidaan mallintaa genotyyppi-fenotyyppi kuvauksena. Piilomuuttujien avulla generaattorin tuottamien pelikenttien joukosta voidaan etsiä tietynlaisia pelikenttiä, esimerkiksi evoluutionaalisella piilomuuttujaoptimoinnilla, joka on esitetty luvussa 5.

5. GENERATIIVISTEN KILPAILEVIEN VERKKOJEN HEIKKOUDET

Tässä luvussa esitetään ensin yleisiä ongelmia, joita esiintyy generatiiviset kilpailevien verkkojen tuottamissa videopelikentissä ja sen jälkeen ratkaisuja näihin ongelmiin. Ratkaisut on eritelty seuraaviin kategorioihin: neuroverkon rakenne, harjoitusdatan manipulaatio, evoluutionaalinen optimointi ja videopelikentän korjaus.

5.1 Generatiiviset kilpailevat verkot ja videopelikentät

Generatiiviset kilpailevat verkot pystyvät onnistuneesti tuottamaan kuvia ja muita samankaltaisia sisältötyyppejä, joilla ei ole rakenteellisia vaatimuksia. Kuitenkin kun puhutaan videopelikentistä, yksinkertaiset GAN-verkot eivät pysty mallintamaan niitä tarpeeksi tarkasti (Torrado ja muut, 2019). Seuraavaksi on esitetty kolme yleistä ongelmaa.

Kenttien pelattavuus

Generatiiviset kilpailevat verkot eivät opi tarpeeksi hyvin videopelikenttien perusteita ja usein tuottavat kenttiä, joita ei voi läpäistä (Torrado ja muut, 2019). Näihin ongelmiin on esitetty useita ratkaisumenetelmiä, joita tarkastellaan seuraavissa aliluvuissa.

Monimuotoisuus

Verkon tuottamien kenttien monimuotoisuus kärsii, jos suuri osa kentistä on mahdotonta läpäistä. Tuotettujen kenttien joukon täytyy sisältää tarpeeksi variaatiota, jotta niitä voitaisiin hyödyntää tehokkaasti videopeleissä.

Globaalit piirteet

Generatiiviset kilpailevat verkot eivät välttämättä ymmärrä globaaleja rakenteellisia ominaisuuksia (Risi ja Togelius, 2019). Kuitenkin globaalien ominaisuuksien tunnistamista on pystytty parantamaan muuttamalla neuroverkon rakennetta esimerkiksi GlobalGAN (Volz, Justesen ja muut, 2020) ja CESAGAN toteutuksissa (Torrado ja muut, 2019).

5.2 Neuroverkon rakenne

Eri arkkitehtuurillisilla malleilla voidaan parantaa tuotettujen videopelikenttien laatua. Malleja on lukuisia, mutta tässä työssä tarkastellaan CESAGAN ja MultiGAN malleja, joilla on pyritty ratkaisemaan ongelmia liittyen pelattavuuteen ja monimuotoisuuteen. Taulukossa 5.1 näkyy miten GAN ei pysty mallintamaan tarpeeksi usein pelattavuuteen liittyviä ominaisuuksia. Tavallinen GAN tuottaa pelattavan kentän vain noin joka viides kerta, kun taas CESAGAN pystyy tuottamaan pelattavia kenttiä 58 %:ssa suorituskerroista

Tuotettujen videopelikenttien laadun parantamiseksi generatiiviset kilpailevat verkot ovat usein syviä ja konvolutionaalisia. Usein käytetään myös edellä mainittua Wassersteinin-harjoitusalgoritmia verkon harjoitusvaiheen tasapainottamiseen.

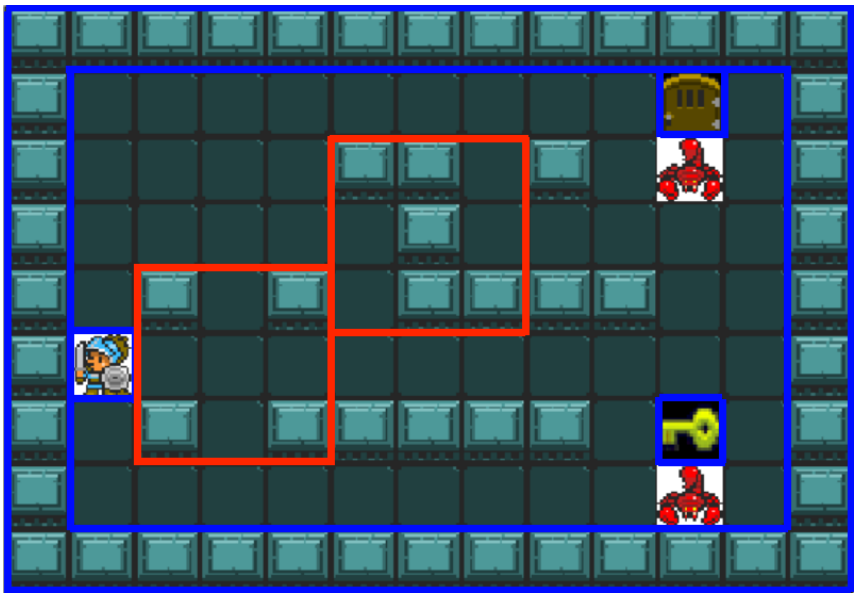
CESAGAN

Torrado ja muut (2019) generoivat uusia videopelikenttiä käyttäen heidän esittämäänsä ehdollisesti upottavaa itsensä-huomioivaa generatiivista kilpailevaa verkkoa (engl. Conditional Embedding Self-Attention Generative Adversarial Network, CESAGAN). He tuottavat kenttiä General Video Game Artificial Intelligence (GVG-AI) ympäristön Zelda-peliin, joka on yksinkertaistettu versio Nintendon julkaisemasta The Legend of Zelda -videopelistä. Zeldassa kentän alitavoitteena on löytää avain ja päätavoitteena avata ovi avaimella. Pelikenttien sisältämät esteen voidaan jakaa vihollisiin ja seiniin.

Konvolutionaalisten GAN:ien käyttö on ongelmallista videopelikenttien yhteydessä. Konvoluution avulla pystytään oppimaan lokaaleja piirteitä tehokkaasti, mutta globaalien piirteiden, kuten esimerkiksi eri puolilla pelikenttää sijaitsevien tavoitteiden oppiminen aiheuttaa ongelmia. Tarpeeksi syvällä konvoluutioneuroverkolla tämä olisi mahdollista, mutta tämä puolestaan aiheuttaa lisää epävakautta ja laskentaa harjoitusvaiheessa. Globaalit piirteet ovat videopelikentän kokonaisuuden kannalta tärkeitä piirteitä, kun taas lokaalit piirteet vallitsevat tietyllä videopelikentän osa-alueella (Volz, Justesen ja muut, 2020). Kuvassa 5.1 on esitetty esimerkkejä Zelda-pelikentän globaaleista ja lokaaleista piirteistä.

Tulokset 45 harjoitusdatakentällä		
Neuroverkkotyyppi	pelattavat kentät	kopioituneet kentät
GAN	19.4 %	39.4 %
CESAGAN	58 %	37.6 %
Tulokset 5 harjoitusdatakentällä		
GAN	24.6 %	98 %
CESAGAN + bootsrap	47 %	60.3 %

Taulukko 5.1. *Zelda peliin tuotettujen pelattavien ja kopioituneiden videopelikenttien suhteet (Torrado ja muut, 2019).*



Kuva 5.1. *Zelda-kentän sinisellä ympyröidyt globaalit ja punaisella ympyröidyt lokaalit piirteet (Torrado ja muut, 2019)*

Kokonaisuuden kannalta tärkeät piirteet ovat kenttää ympäröivä seinä ja pelaajahahmon ja sen tavoitteiden välinen yhteys ja määrä. Lokaaleita piirteitä kuvassa ovat esimerkiksi seinien muodostamat esteet.

Tähän ongelmaan ratkaisuna on käytetty itsensä huomioivaa GAN:ia (SAGAN), jossa konvolutionaaliseen GAN:iin lisätään itsensä huomioiva kerros, jonka tavoitteena on edistää globaalien piirteiden oppimista. Itsensä huomioiva GAN hyödyntää huomiokuvausmatriisia, joka koostuu kolmen ominaisuusvektorin muodostamasta todennäköisyysjakaumasta. Huomiokuvaus sisältää artikkelin sisäiset riippuvuudet, joita painotetaan neuroverkon oppimisessa.

Itsensä huomioivaan kerrokseen voidaan lisätä pelattavuuteen liittyviä ominaisuuksia ehdollisella upotuksella (engl. conditional embedding). Ehdollinen upotuksen avulla voidaan kontrolloida tiettyjä videopelikentän ominaisuuksia. Sillä voidaan esimerkiksi rajoittaa vihollisten tai pelaajahahmojen määrää. Ehdollisessa upotuksessa optimoidaan ominaisuusvektoria. Torrado ja muut (2019) käyttivät ominaisuuksina eri komponenttien lukumääriä, joiden optimointiin käytettiin monikerrosperseptronia.

MultiGAN

Capps ja Schrum (2021) tuottivat kenttäkokonaisuuksia Mega Man -videopeliin (Capcom, 1987). Työssä verrataan yksittäisen GAN:in, OneGAN ja usean GAN:in kokonaisuuden MultiGAN:in tuottamia kenttiä. Mega Man -kentissä uniikkia on niiden rakenne, yksittäinen kenttä koostuu useasta alikentästä, jotka muodostavat käärmemäisen kenttärakenteen. Alikenttiä on seitsemän erilaista: ylä, ala, horisontaalinen ja jokaiselle kulmalle (ylä-

vasen, yläoikea, alavasen, alaoikea) oma. MultiGAN on jaettu niin että jokaiselle alikenttätyypille on oma GAN-verkko, joka pyrkii mallintamaan vain sen tyyppistä videopelikenttää. OneGAN puolestaan pyrkii mallintamaan kaikkia alikenttätyyppejä yhtä aikaa.

Työssä huomattiin MultiGAN-toteutuksen tuottavan pelaajakyselyn perusteella inhimillisempiä ja pelattavampia videopelikenttiä kuin OneGAN. MultiGAN siis osoittaa, että monimutkikkaampienkin videopelien mallintaminen GAN:eilla on mahdollista.

Neuroverkon rakenteella voidaan vaikuttaa tuloksiin huomattavalla tavalla. On kuitenkin huomattava, että sovellukset ovat usein pelikohtaisia ja tarvitsevat ammattitaitoa ja tietämystä GAN:ien toiminnasta, jotta niitä voidaan soveltaa tehokkaasti.

5.3 Harjoitusdatan lisääminen

Generatiivisten kilpailevien verkkojen onnistunut soveltaminen vaatii usein paljon harjoitusdataa. Generaattorin tuottamien kenttien monimuotoisuus paranee mitä enemmän harjoitusdatassa on monimuotoisuutta. Tämä on ongelma videopelikenttien yhteydessä, koska niiden määrä on usein rajoitettu. Rajoitettuun harjoitusdatan määrään koneoppimisen yhteydessä voidaan käyttää esimerkiksi ”bootstrap sampling” -menetelmää, jonka ideana on luoda lisää harjoitusdataa jo olemassa olevasta datasta.

Torrado ja muut (2019) käyttävät onnistuneesti bootstrap -menetelmää harjoitteludatan määrän lisäämiseksi. Generatiivisten kilpailevien verkkojen yhteydessä uusia näytteitä harjoitusdataan voidaan ottaa generaattorin tuottamista näytteistä esimerkiksi seuraavalla pelikohtaisella heuristiikalla:

Zelda-pelin laadukkaiden kenttien heuristiikka

1. On vain yksi pelaajahahmo.
2. On vain yksi avain.
3. On vain yksi ovi.
4. Viholliset peittävät maksimissaan 60 % pelialueesta.
5. Pelaajahahmo voi saavuttaa avaimen sekä oven käyttämällä A*-algoritmia
6. Videopelikenttä sisältää seinäreunuksen, jotta pelaajahahmo ei pääse sen ulkopuolelle.

Heuristiikalle vaihtoehtoinen keino olisi käyttää automaattista pelaaja-agenttia, joka antaa kentälle arvosanan riippuen siitä, miten hyvin se siitä suoriutui. Pelaaja-agentin käyttö kuitenkin lisää verkon harjoitusaikaa.

Uusia kenttiä lisätään harjoitusdatan joukkoon jokaisen harjoituskierroksen - epookin - jälkeen, pois lukien kopioituneet kentät. Torrado ja muut (2019) tuottamien pelattavien ja

kopioituneiden kenttien määrät on esitetty taulukossa 5.1. Bootstrap-menetelmää käytetiin onnistuneesti, koska vain viidellä harjoitusdatakentällä päästiin 47 % pelattavuuteen, joka on vain 11 % huonompi 45 harjoitusdatakentällä tuotetusta tuloksesta.

5.4 Evoluutionaalinen optimointi

Evoluutionaalinen optimointi on tekniikka, jossa näytteitä jalostetaan algoritmisesti haluttuun suuntaan. Bontrager ja muut (2018) kehittämää evoluutionaalisen optimoinnin versiota: piilomuuttujaevoluutiota (engl. latent variable evolution) voidaan käyttää GAN:ien generaattorin tulosavaruuden optimointiin. Työssä luotiin GAN:illa sormenjälkiä, joiden vakuuttavuutta optimoitiin piilomuuttujaevoluutiolla.

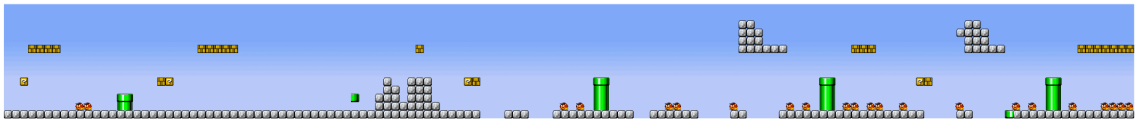
Algoritmi arvostelee ja järjestää sille annetun näytejoukon kelpoisuusfunktion mukaan. Alkuperäisestä joukosta muodostetaan osajoukko arvostelun perusteella, josta voidaan muodostaa uusi näytejoukko eli seuraava sukupolvi. Seuraavaan sukupolveen voidaan ottaa edellisen parhaat yksilöt (elitismi) sekä risteyttää edellisen sukupolven parhaimpia yksilöitä keskenään muodostaen variaatiota. Tämä prosessi toistetaan, kunnes saavutetaan suppeneminen tai jokin ennalta määritelty loppupiste (Bontrager ja muut, 2018).

Kelpoisuusfunktiot

Evoluutionaalisessa optimoinnissa näytteen arvioi kelpoisuusfunktio (engl. fitness function). Kelpoisuusfunktio voi olla matemaattinen funktio tai esimerkiksi interaktiivinen versio, jossa ihmisen mieltymystä kysytään.

Kelpoisuusfunktiot voidaan jakaa kolmeen avainluokkaan: suora, simuloitu ja interaktiivinen. *Suorassa evaluaatiossa* määrittely vastaa toteutusta. Evaluaation kohteena voi olla esimerkiksi sokkelon reittien määrä tai aseen ampumisnopeus. Suorassa evaluaatiossa parametrit ovat usein sovelluskohtaisia. *Simuloitu evaluaatio* hyödyntää agenttia, jonka suorituskyky esimerkiksi videopelikentän läpipeluussa vaikuttaa kentän arvosanaan. Arvosanaan vaikuttavia määreitä voivat olla esimerkiksi: pääsikö agentti kentän läpi, mikä oli läpäisyn suoritusaika tai kuinka paljon agentin pelityylissä löytyi variaatioita. *Interaktiivinen evaluaatio* hyödyntää pelaajan tai suunnittelijan antamaa palautetta pelistä. Pelaajan yhteydessä arvioitavat asiat ovat: kulunut aika tietyissä pelin osissa, milloin pelaaja lopettaa pelaamisen tai näppäinpainalluksien intensiteetti (Togelius ja muut, 2011).

Kelpoisuusfunktiolla voidaan vaikuttaa vahvasti lopullisten yksilöiden piirteisiin, mikä on myös hyödyllistä videopelikenttien yhteydessä.



Kuva 5.2. *Super Mario Bros videopelikenttä, jossa pelikentän vaikeusaste kasvaa vasemmalta oikealle (Volz, Schrum ja muut, 2018)*

Piilomuuttujaevoluutio ja videopelikentät

Volz, Schrum ja muut (2018) työssä käytettiin piilomuuttujaevoluutiota ensimmäisen kerran videopelikenttien yhteydessä. Työssä optimoitiin GAN:in tuottamien Super Mario Bros -kenttien vaikeusastetta. Vaikeusastetta pyrittiin muokkaamaan vihollisten ja tasohyppelyesteiden määrien avulla. Työssä huomattiin piilomuuttujaevoluution toimivan hyvin videopelikenttien eri osa-alueiden optimoinnissa. Kelpoisuusfunktiona käytettiin kenttien staattisia ominaisuuksia (esim. vihollisten määrää) ja simuloitua pelaaja-agenttia.

Kuvassa 5.2 esitetty Super Mario Bros pelikenttä havainnollistaa miten vaikeusastetta on pystytty kontrolloimaan piilomuuttujaoptimoinnilla. Vaikeusaste kasvaa vähitellen oikealle mentäessä. Volz, Schrum ja muut (2018) ehdottavat, että tämänkaltainen prosessointi voisi tulevaisuudessa mahdollistaa dynaamisen vaikeusasteen, jossa pelikentän vaikeusastetta muokataan reaaliajassa pelaajan taitotason mukaan. Dynaaminen vaikeusaste on yksi keino maksimoida pelaajan pelikokemus vaikeusasteen kannalta.

Capps ja Schrum (2021) optimoivat piilomuuttujaevoluutiolla useita eri Mega Man -kenttien määreitä. Määreinä olivat A*-polunetsintäalgoritmin löytämän reitin pituus ja kenttien kytkeytyvyys. Reitin pituus on pelikentän alku ja loppupisteen välinen matka, ja jos A* ei sitä löydä annetaan arvosanaksi -1. Tällä pyritään estämään läpäisemättömät kentät. Kytkeytyvyys (engl. connectivity) määritellään saavutettavien ja saavuttamattomien laattojen suhteena. Kytkeytyvyydellä pyritään minimoimaan pelikenttien turha tila ja näin ollen tekemään kentistä inhimillisempiä. Työssä huomattiin että kenttien sujuvuutta pystyttiin parantamaan geneettisellä optimoiniilla.

Evoluutionaalisella optimoinnilla voidaan jalostaa GAN:ien tuottamaa näytejoukkoa haluttuun suuntaan, mikä on hyödyllistä, sillä GAN:in näytejoukon oppiminen on harvoin täydellistä, varsinkin videopelikenttien yhteydessä.

5.5 Videopelikentän korjaus

Luo ja korjaa -menetelmissä annetaan GAN:in tuottaa videopelikenttiä ilman erityisiä rajoitteita. Jos videopelikentän huomataan olevan mahdoton läpäistä tai muuten pelin sääntöjen vastainen voidaan tämä korjata käyttäen esimerkiksi algoritmeja tai neuroverkkoja.

Neuroverkkotyyppi	pelattavat kentät	kopioituneet kentät	pelattavat ja uniikit kentät
GAN	24.3 %	46.9 %	12.9 %
GAN + MIP	100 %	14.9 %	85.1 %

Taulukko 5.2. *Zelda peliin tuotettujen pelattavien ja kopioidujen videopelikenttien suhteet (Zhang ja muut, 2020)*

Lineaarinen kokonaislukuoptimointi

Zhang ja muut (2020) esittävät luo ja korjaa -ratkaisun automaattiseen videopelikentän tuottamiseen Pac-Man:iin ja jo edellä mainittuun yksinkertaistettuun Zeldaan. Työssä käytetty neuroverkkoarkkitehtuuri on syvä konvolutionaalinen generatiivinen kilpaileva verkko eli DCGAN, jonka tuottamia kenttiä korjattiin lineaarisella kokonaislukuoptimoinnilla (engl. Mixed-Integer linear Program, MIP). Tällä lähestymistavalla pyritään tuottamaan videopelikenttiä, jotka täyttävät pelattavuusrajoitukset ja ovat ulkonäöltään samankaltaisia ihmisten tuottamiin kenttiin verrattuina. Luo ja korjaa -menetelmän menestyminen pohjautuu GAN:in kykyyn ymmärtää inhimillisiä rakenteellisia piirteitä ja MIP:in kykyyn korjata nämä kentät pelattaviksi.

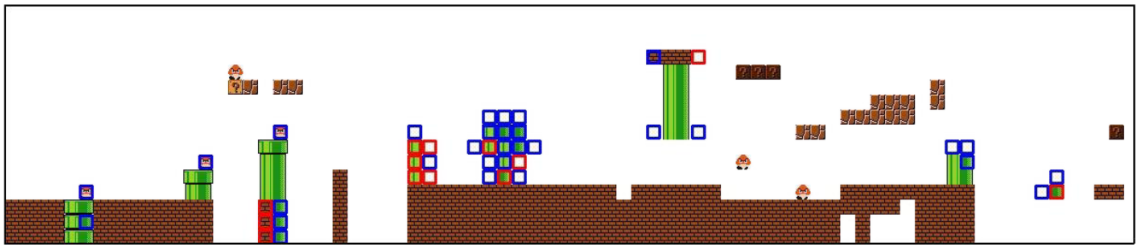
Jokainen diskreetti kaksiulotteinen videopelikenttä voidaan mallintaa tilagraafina. Graafi sisältää solmuja, jotka sisältävät jonkin peliobjektin. Zelda -pelin komponentit on esitetty taulukossa 2.2. Solmut ovat kytketty viereisiin solmuihin, mitkä yhdessä muodostavat tilagraafin.

Lineaarisella kokonaislukuoptimoinnilla voidaan mallintaa videopelikentän rajoitteita. Rajoitteet voivat olla joko yleisiä tai pelikohtaisia. Zhang ja muut (2020) esittävät seuraavat rajoitteet toteutukselleen:

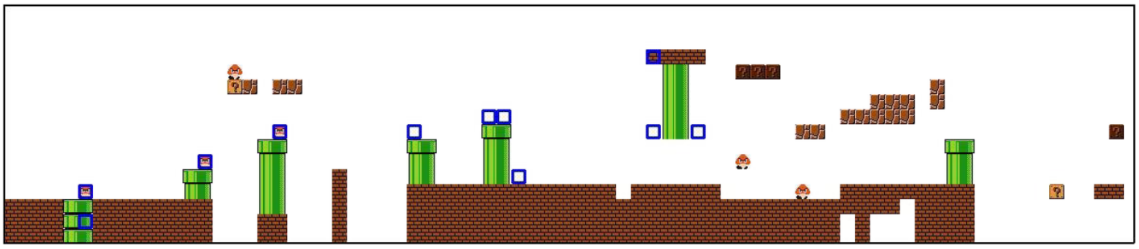
1. Solmun ainutlaatuisuus: Solmu sisältää vain yhden peliobjektin.
2. Saavutettavuus: Lähdeobjektit pystyvät saavuttamaan kaikki maaliobjektit.
3. Muut pelikohtaiset rajoitteet, kuten Zeldaan luvussa 5.3 esitelty heuristiikka 5.3

Saavutettavuus voidaan mallintaa virtausongelmana, jossa lähdeobjektien S täytyy saavuttaa maaliobjektit T. Esimerkiksi Zelda pelissä pelaajahahmon tavoitteet eli avain ja ovi täytyy olla saavutettavissa ja tavoitteiden välissä ei saa olla läpäisemättömiä objekteja kuten seiniä. Zhang ja muut (2020) minimoivat korjauksien määrän ja antavat erityyppisille korjauksille eri kustannukset. On siis tehokkaampaa, jos korjauksessa siirretään peliobjekti viereiseen paikkaan, kuin poistaa ja sijoittaa se uudelleen.

Taulukossa 5.2 verrataan GAN ja GAN + MIP-tekniikoita pelattavuuden ja kenttien monimuotoisuuden kautta. Koska luo ja korjaa -menetelmällä tuotetut videopelikentät ovat aina läpäistävissä, generoitujen kenttien monimuotoisuus säilyy ja läpäisemättömiä kenttiä ei tarvitse hylätä. Pelattavien sekä uniikkien kenttien määrä tuotettujen kenttien joukosta on



Kuva 5.3. *Generatiivisen kilpailevan verkon tuottama Super Mario Bros kenttä, jonka CNET:in mukaan vialliset ja epävakaat laatat on merkitty vastaavasti punaisella ja sinisellä (Shu ja muut, 2020).*



Kuva 5.4. *Paras korjaus 25 generaation jälkeen (Shu ja muut, 2020).*

GAN + MIP menetelmällä 85.1 %, kun taas tavallisella GAN-toteutuksella vain 12.9 %.

Luo ja korjaa voidaan myös verrata CESAGAN-toteutukseen (Torrado ja muut, 2019), jossa pelattavien kenttien määrä oli vain 58 % verrattuna GAN:in ja MIP:in yhdistelmän 100 %:iin. Kopioituneiden kenttien määrä on myös pienempi GAN + MIP-toteutuksessa - vain 14.9 %, kun taas CESAGAN:in tuottamista kentistä 37.6 % ovat toistensa kopioita. Luo ja korjaa -menetelmä siis mallintaa Zelda-pelin kenttiä näiden määreiden mukaan paremmin kuin CESAGAN-menetelmä.

Luo ja korjaa -menetelmää verrattiin myös esteettisellä tasolla tavallisen GAN:in ja ihmisten tekemiin kenttiin. Estetiikkaa mitattiin tyhjien laattojen, seinien ja vihollisten määrän sekä kentän ratkaisemispolun pituuden avulla. Ratkaisemispolun pituus on parempi olla suuri, koska Zelda-kentät ovat silloin mielenkiintoisempia. Työssä huomattiin, että verrattuna tavalliseen GAN-toteutukseen GAN:in ja MIP:in yhdistelmä tuottaa parempia kenttiä myös esteettisellä tasolla.

Lineaarinen kokonaislukuoptimointi sisältää kuitenkin rajoituksia. Vaikka MIP:it ovat tehokkaita mallinnustyökaluja, tarvitaan niiden soveltamiseen ammattitaitoa. MIP:ien soveltaminen kaikkiin videopeleihin ei ole myöskään suoraviivaista ja sen soveltaminen monimutkaisempiin peleihin pelaajahahmon liikkeen kannalta, kuten Super Mario Bros tarvitsee lisätutkimusta.

CNET-korjaus

Shu ja muut (2020) esittävät työssään neuroverkkokorjaajan CNET:in, joka pystyy korjaamaan viallisia elementtejä videopelikentissä. Korjaaja luokittelee pelikentän laattoja vialliksiksi niiden ympäristön (esimerkiksi yhden laatan etäisyydellä) perusteella ja pystyy ehdottamaan laatan tilalle uuden paremman laatan. Työssä korjattiin onnistuneesti GAN:in tuottamia viallisia Super Mario Bros -kenttiä, mutta sitä pystyttäisiin soveltamaan myös muihin laattapohjaisiin videopeleihin.

Pelkästään CNET:in käyttö videopelikenttien korjaamiseen ei ole tarpeeksi tarkka ja vialliset laatat aiheuttavat ympäristöönsä häiriöitä. Korjausta optimoitiin geneettisellä algoritmilla, jossa minimoitiin virheiden ja muutettujen laattojen määrää. Kuvassa 5.3 on esitetty GAN:in tuottama Super Mario Bros -pelin kenttä, joka on 25:n sukupolven geneettisen optimoinnin jälkeen esitetty kuvassa 5.4. Korjaaja onnistuu korjaamaan vialliset putkirakenteet, jotta kenttä näyttäisi inhimillisemmältä.

6. YHTEENVETO JA POHDINTAA

Tutkielman tavoitteena oli tutkia, miten generatiivisilla kilpailevilla verkoilla tuotettujen videopelikenttien pelattavuuteen, ulkonäköön ja vaikeusasteeseen liittyvät määreet on ratkaistu. Tutkielmassa tarkasteltiin ratkaisuja ongelmiin neljällä eri osa-alueella: neuroverkon rakenne, harjoitusdatan lisääminen, evoluutionaalinen optimointi ja videopelikentän korjaus. Generatiivisten kilpailevien verkkojen kyky mallintaa inhimillisiä ulkonäöllisiä piirteitä on niiden vahvuus, mutta videopelikenttien rakenteelliset pelattavuuteen liittyvät vaatimukset aiheuttavat usein ongelmia, minkä takia pelkkä GAN ei usein riitä. Apukeinoja GAN:ien yhteydessä ovat esimerkiksi: erilaiset GAN:in rakenteelliset variaatiot, kuten CE-SAGAN ja MultiGAN, geneettinen optimointi ja eri korjausmenetelmät, kuten kokonaislukuoptimointi.

Videopelikenttiä on arvioitu esimerkiksi monimuotoisuuden ja ulkonäköön liittyvien määreiden avulla, mitkä eivät välttämättä vastaa täysin totuutta (Torrado ja muut, 2019) (Zhang ja muut, 2020). Paras keino olisi arvioida videopelikenttiä ihmisten avulla, mutta näytteiden määrän ollessa suuri tämä on mahdotonta.

Optimaalitilanne pelikenttien tuottamiselle olisi mahdollistaa eri ratkaisujen objektiivinen yleinen vertailu. Tämä ei kuitenkaan ole tällä hetkellä mahdollista, sillä peliympäristön muututtua ei mikään välttämättä toimi samalla tavalla. Mahdollinen ratkaisu eri pelien kenttien arviointiin olisi käyttää yleistä pelaaja-agenttia, joka pystyisi toimimaan eri pelimaailmoissa ja arvioimaan niitä (Risi ja Togelius, 2019), mutta tämä vaatii vielä paljon lisätutkimusta.

Koneoppivilla menetelmillä kuten GAN:eilla pyritään mahdollistamaan resurssiystävällinen keino tuottaa videopelikenttiä. Tästä ihanteesta ollaan kuitenkin vielä kaukana, sillä yksinkertaistenkin pelien tuottaminen aiheuttaa ongelmia. Ehkä tällä hetkellä todennäköisempi lähestymistapa on hyödyntää koneoppivia malleja ihmisten kanssa, jolloin esimerkiksi GAN:ilta voidaan saada inspiraatiota videopelikenttien mallintamiseen. Suunnittelija pystyisi muokkaamaan verkon tuottamaa tasoa halutessaan, jos siinä on jotain vialla.

Tulevatko koneoppimisen sisällöntuotannon keinot syrjäyttämään perinteiset algoritmiset ratkaisut? Koneoppivien keinojen satunnaisuus ja epävakaus aiheuttavat sen, että niitä on vaikea hyödyntää pelikehityksessä. Proseduraalinen sisällöntuotanto on tällä hetkellä paljon turvallisempi ja vakuuttavampi keino luoda videopelikenttiä. Koneoppisen metodit

toimivat myös paremmin mitä enemmän dataa on olemassa ja uutta peliä luotaessa ei dataa tietenkään ole. Tämä ei kuitenkaan tarkoita sitä, että toinen olisi parempi kuin toinen. PCG ja PCGML ovat eri menetelmiä jotka soveltuvat eri tilanteisiin. On myös mielenkiintoista huomata se, että PCG-vaikutteisia metodeja on lähiaikoina alettu käyttämään koneoppivien mallien parantamiseen (Risi ja Togelius, 2019).

Koska koneoppimisen ja varsinkin syväoppimisen tieteenala ovat vaikeasti ennustettavissa on mahdoton sanoa, miltä huipputason metodit näyttävät tulevaisuudessa.

LÄHTEET

Arjovsky, Martin, Soumith Chintala ja Léon Bottou (2017). ”Wasserstein generative adversarial networks”. eng. Teoksessa: *34th International Conference on Machine Learning, ICML 2017*. Vol. 1, s. 298–321. ISBN: 1510855149.

Audevart, Alexia, Konrad Banachewicz ja Luca Massaron (2021). *Machine Learning Using TensorFlow Cookbook: Create Powerful Machine Learning Algorithms with TensorFlow*. eng. Birmingham: Packt Publishing, Limited, s. 237–239. ISBN: 1800208863.

Bjork, Staffan ja Jussi Holopainen (2005). *Patterns in game design*. eng. Hingham (Mass.): Charles River Media, s. 55–105. ISBN: 1-58450-354-8.

Bontrager, Philip, Aditi Roy, Julian Togelius, Nasir Memon ja Arun Ross (2018). ”DeepMasterPrints: Generating masterprints for dictionary attacks via latent variable evolution”. eng. Teoksessa: *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems, BTAS 2018*. ISBN: 1538671794.

Byrne, Edward (2005). *Game level design*. Hingham (Mass.): Charles River Media, s. 55–74.

Capps, Benjamin ja Jacob Schrum (2021). ”Using multiple generative adversarial networks to build better-connected levels for mega man”. eng. Teoksessa: *GECCO 2021 - Proceedings of the 2021 Genetic and Evolutionary Computation Conference*, s. 66–74. ISBN: 9781450383509.

Davoodi, Omid, Mehrdad Ashtiani ja Morteza Rajabi (2020). ”An Approach for the Evaluation and Correction of Manually Designed Video Game Levels Using Deep Neural Networks”. eng. *Computer journal*. ISSN: 0010-4620.

Goodfellow, Ian J, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville ja Yoshua Bengio (2014). ”Generative Adversarial Networks”. eng. DOI: 10.48550/ARXIV.1406.2661. URL: https://arxiv.org/abs/1406.2661.

Liu, Jialin, Sam Snodgrass, Ahmed Khalifa, Sebastian Risi, Georgios N Yannakakis ja Julian Togelius (2020). ”Deep learning for procedural content generation”. eng. *Neural computing applications* 33.1, s. 19–37. ISSN: 0941-0643.

Risi, Sebastian ja Togelius (2019). ”Increasing Generality in Machine Learning through Procedural Content Generation”. eng. DOI: 10.48550/ARXIV.1911.13071. URL: https://arxiv.org/abs/1911.13071.

Shaker, Noor (2016). *Procedural Content Generation in Games*. eng. Computational synthesis and creative systems. Cham: Springer International Publishing AG. ISBN: 9783319427140.

Shu, Tianye, Ziqi Wang, Jialin Liu ja Xin Yao (2020). ”A Novel CNet-assisted Evolutionary Level Repairer and Its Applications to Super Mario Bros”. eng. Teoksessa: *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, s. 1–10. ISBN: 1728169291.

Smith, Gillian, Mee Cha ja Jim Whitehead (2008). ”A framework for analysis of 2D platformer levels”. eng. Teoksessa: *Proceedings of the 2008 ACM SIGGRAPH symposium on video games*. Sandbox ’08. ACM, s. 75–80. ISBN: 1605581739.

Summerville, Adam, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgard, Amy K Hoover, Aaron Isaksen, Andy Nealen ja Julian Togelius (2018). ”Procedural Content Generation via Machine Learning (PCGML)”. eng. *IEEE transactions on games* 10.3, s. 257–270. ISSN: 2475-1502.

Togelius, G. N Yannakakis, K. O Stanley ja C Browne (2011). ”Search-Based Procedural Content Generation: A Taxonomy and Survey”. eng. *IEEE transactions on computational intelligence and AI in games.* 3.3, s. 172–186. ISSN: 1943-068X.

Torrado, Ruben Rodriguez, Ahmed Khalifa, Michael Cerny Green, Niels Justesen, Sebastian Risi ja Julian Togelius (2019). ”Bootstrapping Conditional GANs for Video Game Level Generation”. eng. DOI: 10.48550/ARXIV.1910.01603. URL: https://arxiv.org/abs/1910.01603.

Volz, Vanessa, Niels Justesen, Sam Snodgrass, Sahar Asadi, Sami Purmonen, Christoffer Holmgard, Julian Togelius ja Sebastian Risi (2020). ”Capturing Local and Global Patterns in Procedural Content Generation via Machine Learning”. eng. Teoksessa: *IEEE Conference on Computatonal Intelligence and Games, CIG*. Vol. 2020-, s. 399–406. ISBN: 1728145333.

Volz, Vanessa, Jacob Schrum, Jialin Liu, Simon Lucas, Adam Smith ja Sebastian Risi (2018). ”Evolving mario levels in the latent space of a deep convolutional generative adversarial network”. eng. Teoksessa: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO ’18. ACM, s. 221–228. ISBN: 1450356184.

Zhang, Hejia, Matthew C Fontaine, Amy K Hoover, Julian Togelius, Bistra Dilkina ja Stefanos Nikolaidis (2020). ”Video Game Level Repair via Mixed Integer Linear Programming”. eng. DOI: 10.48550/ARXIV.2010.06627. URL: https://arxiv.org/abs/2010.06627.