Toni Hautaoja

Oppiva tekoäly moderneissa videopeleissä

Tietotekniikan kandidaatintutkielma

29. toukokuuta 2019

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Toni Hautaoja

Yhteystiedot: toni.n.a.hautaoja@student.jyu.fi

Ohjaaja: Antti-Jussi Lakanen

Työn nimi: Oppiva tekoäly moderneissa videopeleissä

Title in English: Learning AI in modern video games

Työ: Kandidaatintutkielma

Sivumäärä: 31+0

Tiivistelmä: Tässä tutkimuksessa käsitellään videopeleissä olevia tekoälyhahmoja ja niiden simuloitua oppimista, tähän liittyviä toteutustapoja, käsitteitä sekä videopeleissä hyödynnettävää koneoppimista. Työn tavoitteena on avata näitä asioita erilaisten esimerkkien ja teorian kautta sekä selittää, kuinka tekoälyllistä oppimista on toteutettu moderneissa videopeleissä. Tekoäly ja etenkin koneoppiminen ovat videopelitekoälyhahmoihin toteutettuna vielä melko varhaisella tasolla, eikä varsinaista koneoppimista olla yleisesti käytetty niiden toiminnan ohjaajana. Pelihahmot voivat silti oppia monella eri tavalla. Videopelien kompleksisuuden kasvaminen, teknologian kehittyminen ja jatkuva tiivis tekoälytutkimus mahdollistavat suuren potentiaalin erilaisten uusien, immersiivisten ja realististen videopelikokemusten luomiseksi tulevaisuudessa.

Avainsanat: tekoäly, videopelit, NPC, simuloitu oppiminen, koneoppiminen, vahvistettu oppiminen

Abstract: In this study artificial intelligence (AI) and simulated learning of video game characters, the different ways these concepts are implemented and the use of machine learning are covered in the context of video games. The goal of the study is to explain via different examples and theoretical background how learning through AI is accomplished in the modern video games. When implemented into the non-player-characters in games, AI and especially machine learning are still in quite an early stage and the latter has not been generally part of the actual game character

AIs. Still, video game characters can learn in various of different ways. The increase in the video game complexity, technological advancement and continuous intense research in the AI frontier allow great potential and possibilities for the creation of most diverse, immersive and realistic new gaming experiences in the future.

Keywords: artificial intelligence, video games, NPC, simulated learning, machine learning, reinforcement learning

Kuviot

Sisältö

1 Johdanto

Tässä tutkielmassa käsitellään tietokoneen ohjaamien ei-pelaajahahmojen (engl. non-player character, NPC) tekoälyllistä toimintaa, oppimista ja sitä kuinka nämä pelihahmot saadaan tuntumaan älykkäiltä. Tämän lisäksi tarkastellaan pelihahmojen toteutuksen taustalla piileviä algoritmeja ja koneoppimista. Huomautettakoon, että tutkielman kontekstissa puhuttaessa koneoppimisesta tarkoitetaan koneoppimisalgoritmeihin perustuvaa tekoälyllistä oppimista, kun taas simuloidulla oppimisella tarkoitetaan jotain sellaista pelihahmojen tekoälyyn toteutettua joukkoa algoritmeja ja erilaisia toimintamalleja, joilla voidaan toteuttaa oppimista, mutta jotka eivät pohjaudu koneoppimisalgoritmeihin. Tutkielman kontekstissa tekoälyllinen oppiminen käsittää sekä koneoppimisen että simuloidun oppimisen.

Tutkielman aihe rajattiin videopelihahmojen tekoälyyn, koska jatkuvasti suosiotaan kasvattavat videopelit ovat tekoälytutkimuksen kannalta monia mielenkiintoisia haasteita sisältävä tutkimusala (Szita 2012; Erdem & Halici 2016). Videopelitekoälyä ja siihen liittyviä erilaisia tekoälyllisiä ratkaisuita tarkastellaan tietokoneen ohjaamien pelihahmojen toiminnan, simuloidun oppimisen ja koneoppimisen kannalta modernien videopelien kontekstissa. Moderneilla videopeleillä tarkoitetaan pelejä, jotka pitävät sisällään moninaisia erilaisia haasteita ja mekaniikkoja, joiden puitteissa pelaaja ja pelihahmot kykenevät tekemään erilaisia monimutkaisia toimintoja ja tehtäviä sekä toimivat kanssakäymisessä toistensa ja pelimaailman kanssa. Tutkielmassa syvennytään tarkemmin ei-pelaaviin NPC-vihollishahmoihin ja selitetään kuinka nämä voivat muuttaa käyttäytymistään ja oppia erilaisia toimintamalleja pelaajan tekojen ja pelityylin pohjalta.

Tutkielman tutkimusmenetelmä on kirjallisuuskartoitus. Aihe soveltuu hyvin kirjallisuuskartoituksena tehtäväksi, koska videopeleissä olevasta tekoälystä löytyy paljon tutkimusta ja informaatiota, jota on hyvä analysoida ja koostaa koherentimmaksi ja selkeämmäksi kokonaisuudeksi mahdollista myöhempää ja tarkempaa tutkimusta varten. Tutkielman tieteelliset lähteet on valittu tieteellisten julkaisijoiden ja julkaisujen, kuten vertaisarvioitujen tutkimus- ja lehtiartikkelien sekä kirjojen pohjalta.

Tieteellisten lähteiden tukena on käytetty lisäksi muita lähteitä täydentämään tutkielmassa käsiteltävää asiaa ja siinä esiintyviä esimerkkejä. Tieteellisen lähdemateriaalin hakukoneina toimivat JykDok, Julkaisufoorumi, IEEE Xplore ja Google Scholar. Lähteet sijoittuvat muutamaa poikkeusta lukuun ottamatta viimeisen 15 vuoden ajalle.

Tutkielman aihetta käsitellään ensin selittämällä luvussa 2 tutkielman teoriataustan kannalta tärkeät käsitteet paneutuen siihen, mitä ovat tekoäly ja koneoppiminen, vahvistettu oppiminen ja tähän liittyvä Q-learning -algoritmi. Luvussa 3 käsitellään aihetta tarkemmin videopelikontekstissa tutkimalla ensin NPC-hahmoja ja niiden toimintaa, josta siirrytään pohtimaan koneoppimisen käyttöä videopeleissä. Tämän jälkeen tutkielma pyrkii antamaan tekoälyvihollisten oppimisen simuloimisesta käytännönläheisen malliesimerkin käyttämällä apunaan muutamaa konkreettista videopeliesimerkkiä. Luvussa 4 kootaan tutkielman tärkeimmät kohdat ja johtopäätökset yhteen sekä pohditaan hieman videopelitekoälyn tulevaisuutta ja siellä odottavia mahdollisuuksia, haasteita ja ongelmia, joita pitäisi ratkoa realistisemman ja fiksumman videopelitekoälyn löytämiseksi.

2 Keskeiset käsitteet ja taustat

Tässä kappaleessa avataan tutkielman kannalta tärkeiden käsitteiden tarkoitusperää ja teoriataustaa aloittamalla kaiken pohjana toimivasta tekoälystä ja koneoppimisesta, jonka jälkeen siirrytään syvemmin vahvistettuun koneoppimisen ja siihen liitetyn Q-learning -algoritmin toteutuksen ja toiminnan tarkasteluun.

2.1 Tekoäly

Tekoälyksi (engl. artificial intelligence, AI) kutsutaan sellaista tietojenkäsittelytieteen osa-aluetta, joka keskittyy älykkäiden entiteettien, kuten koneiden ja ohjelmien tai ohjelmistojen ymmärtämiseen ja luomiseen (Russell & Norvig 2016, s. 1). Russellin ja Norvigin (2016, s. 34) mukaan tekoäly tutkii älykkäitä agentteja, jotka havainnoivat erilaisia ympäristöjä ja toteuttavat niihin perustuen erilaisia toimintoja. Tekoälyn tarkoituksena on pyrkiä matkimaan ihmisen tajuntaa, mieltä ja toimintamalleja ja suoriutua erinäisistä tehtävistä kuten ihmisillä on tapana (Russell & Norvig 2016, s. 2). Tekoäly voidaan siis ymmärtää rationaalisesti toimivan koneen tai ohjelman kykynä ajatella ja oppia.

Russell ja Norvig (2016, s. 1020) jakavat tekoälyn **heikkoon ja vahvaan** tekoälyyn, jotka ovat tämän tutkielman kannalta olennaisia käsitteitä tekoälyn laajan skaalan ymmärtämiseksi. **Heikko tekoäly** perustuu siihen, että laitteet ja ohjelmat saadaan simuloitua käyttäytymään älykkäästi (Kaplan 2016, s. 68). Tästä esimerkkinä voidaan käyttää vaikkapa erilaisia shakkitietokoneita (Russell & Norvig 2016, s. 1022). Heikko tekoäly voidaan ymmärtää myös niin kutsuttuna **täydellisen informaation** pelien tekoälynä, jotka ovat olleet ensimmäisiä suuria askelia tekoälyllisessä kehityksessä (Schaeffer & Van den Herik 2002). Kaplan (2016, s. 82) ei oleta heikolla tekoälyllä olevan minkään asteista tietoisuutta.

Vahva tekoäly puolestaan on mieleen verrattava (Kaplan 2016, s. 68) tietoinen voima, joka kykenee itsenäiseen ja intentionaaliseen ajatteluun (Russell & Norvig 2016, s. 1026). Vahvan tekoälyn ja koneen ajattelukyvyn testaamista varten on kehitetty

muutamia erilaisia malleja, joista klassisena esimerkkinä toimii Alan Turingin luoma, myöhemmin **Turingin testinä** tunnettu älykästä käyttäytymistä mittaava koe (1950), jossa testataan pystyykö ihmistarkkailija erottamaan keskusteleeko tämä ihmisen vai koneen kanssa. Alkuperäisen Turingin testin päälle on myöhemmin rakennettu myös niin kutsuttu **Winogradin skeema** (Levesque 2011). Turingin testi on saanut kuitenkin paljon rankkaa kritiikkiä muun muassa filosofi John Searlelta, joka pitää koetta samaan aikaan liian löyhänä ja rajoittavana perustana tietokoneen älykkäälle käyttäytymiselle niin kutsutussa **kiinalaisen huoneen argumentissaan** (1980). Ihmisälyä jäljittelevää tietoisuuden omaavaa ja älykästä tekoälyä ei vielä ole missään muodossa kehitetty (Yampolskiy & Fox 2013).

2.2 Koneoppiminen

Koneoppiminen (engl. machine learning, ML) on tärkeä tekoälyn haara, jota hyödyntävä ohjelma tai ohjelmisto pystyy Kapitanovan ym. (2012) ja Mohammedin (2017) mukaan kokoelmalla erilaisia algoritmeja oppimaan itsenäisesti mukautumalla muutoksiin ja päätymään haluttuun lopputulokseen luodun opetteludatan pohjalta, jossa tiedetään syöte eli alkutila ja haluttu lopputulos. Siten tätä dataa käyttämällä luodaan algoritmi, joka päätyy samaan lopputulokseen kaikilla samankaltaisilla syötteillä (Kapitanova & Sang 2012). Alpaydin (2016, s. 24-25, 35-42) määrittelee koneoppimisen tietokoneiden kykynä optimoida suoriutumistaan käyttäen apuna esimerkkidataa tai aikaisempaa kokemusta. Koneoppiminen ilman erillistä valvontaa vaatii siis ohjelmalta kyvyn tunnistaa erilaisia kaavoja sille syötetyssä datassa, josta se tekee omat johtopäätöksensä.

Yannakis ja Togelius (2018, s. 30), sekä Neittaanmäki ym. (2018) jakavat koneoppimisen kolmeen kategoriaan, joiden pohjalta tähän liittyviä metodeja yleensä käsitellään: erilaisten syöte-tavoite-parien kanssa täysin luokitellun aineiston, kuten kuvien avulla tapahtuva **ohjattu oppiminen** eli supervised learning (Sebastiani 2002), positiiviseen ja negatiiviseen palautteeseen pohjautuva **vahvistettu oppiminen** reinforcement learning sekä **ohjaamaton oppiminen** eli unsupervised learning, joka tapahtuu ihmisen ajattelun kaltaisesti erilaisten representaatioiden sisäisten rakennel-

mien pohjalta ilman valmiiksi luokiteltua opetusaineistoa tunnistamalla opettamiseen käytettävästä datasta riippuvuuksia, suhteita ja eri syötteiden välille aiheutuvia samankaltaisuuksia (Alpaydin 2016; Neittaanmäki ym. 2018). Luokat etsitään ohjaamattomassa oppimisessa datasta luonnollisten ryhmien ja klusterien perusteella (Jain ym. 2000).

2.3 Vahvistettu oppiminen

Vahvistetun oppimisen toiminta perustuu Yannakisin ja Togeliuksen (2018, s. 71) sekä Ducin (2008) mukaan palkitsemiseen behavioristisen palkkio-toru-analogian mukaisesti. Vahvistetussa oppimisessa opetettava kohde eli agentti lähtee suorittamaan prosessia ilman minkäänlaisia lähtötietoja tai päämäärää ja saa näiden sijaan toiminnoistaan palautetta toimintaympäristössään joko palkitsemisena hyvin ja tavoitteiden mukaan tai nuhteina virheiden sattuessa (Russell & Norvig 2016, s. 830-831). Molemmissa tapauksissa oppiminen vahvistuu palkkioiden osoittamaan suuntaan ja ohjaa erilaisten päätösten tekemisessä (Yannakakis & Togelius 2018, s. 71-72). Klassisen ohjatun koneoppimisen tapaan malliesimerkit ja luokiteltu aineisto eivät yleensä ole saatavilla, vaan oppiminen tapahtuu yrittämisen, palautteen ja kokemuksen kautta (Sutton & Barto 2018, s. 2).

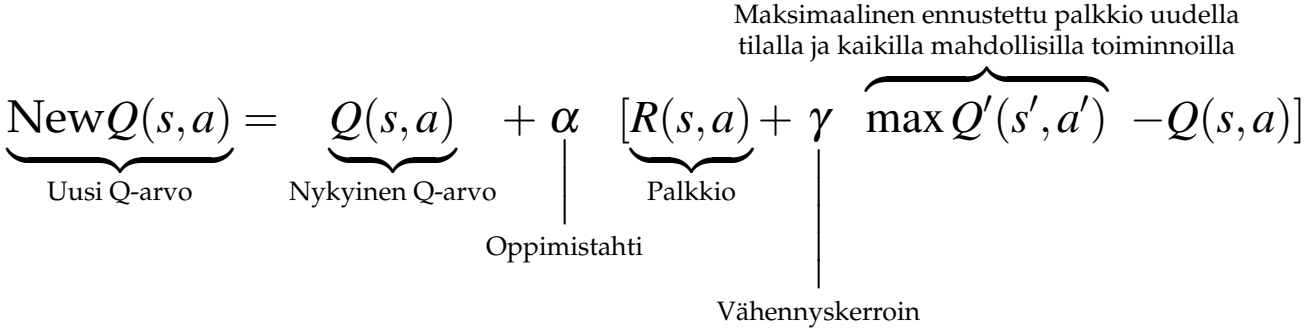
Oppimisessa käytetään pohjana niin kutsuttua **Markovin päätöksentekoprosessia** (Markov Decision Process, MDP), jonka mukaisesti pyritään suorittamaan toimintoja ja etsimään sellainen optimaalinen toimintatapa π, jonka valinnat maksimoivat halutut pitkän aikavälin tavoitteet (Yannakakis & Togelius 2018, s. 72). Optimaalisessa lopputuloksessa jatkuva erilaisten tilanteiden toisto ja näistä saatava palaute opettaa agentin toimimaan ympäristössään tavoitteiden mukaisesti ja halutunlaisesti mahdollisimman tehokkaasti.

Yannakis ja Togelius (2018, s. 74) jakavat vahvistetun oppimisen kolmeen eri algoritmikategoriaan. **Dynaamiseen ohjelmointiin** (engl. dynamic programming) perustuvat algoritmit, jotka edellyttävät erilaista lähtöinformaatiota toimintaympäristöstä ja jossa optimaalinen menettelytavan laskeminen käyttää apunaan niin kutsuttua

"**boostrappingia**" eli suoritetuista toiminnoista saatavaa jatkuvaa palautetta, jonka pohjalta voidaan rekursiivisesti arvioida, päätellä ja päivittää halutun optimaalisen lopputuloksen suuntaa niin, että kaikki aiempi palaute vaikuttaa tuleviin toimintoihin. Dynaamista ohjelmointia vastoin seuraavat kaksi algoritmia voidaan kummatkin toteuttaa ilman minkäänlaisia lähtöarvoja tai tietoa toimintaympäristöstä ja tavoitteista: **Monte Carlo** -algoritmit, joissa tehtävää suorittava agentti saa palkkion vasta suorittaessaan halutun tavoitteen onnistuneesti käyttämättä apunaan bootstrappingia, sekä **TP**- eli **Temporal-Difference** -algoritmit, jotka puolestaan käyttävät apunaan bootstrappingia.

2.4 Q-oppiminen

Yannakisin ja Togeliuksen (2018, s. 74) mukaan **Q-oppiminen** eli **Q-learning** on suosituin ja laajimmin videopelitekoälytutkimuksessa käytetty vahvistetun oppimisen piiriin kuuluva TP-metodi. Algoritmi saa nimensä Kuvio 1:n (Comi 2018) yhtälössä olevasta niin kutsutusta **Q-funktiosta** ja siinä olevista Q-arvoista Q(s, a), jotka tarkoittavat tietyn toiminnon qualitya eli laatua tietyssä tilassa s (Watkins 1992). Jos opetettava agentti on esimerkiksi ajassa t tilassa s, suorittaa tässä toiminnallisuuden a ja saa tästä negatiivista palautetta, mutta suorittaessaan samassa tilassa toiminnallisuuden b palaute onkin positiivista, voidaan Q-funktion arvon olettaa olevan korkeampi ja näin toivotumpi toiminnallisuudella b.



Kuvio 1: Bellmanin yhtälöön pohjautuva Q-funktio, jonka pohjalta Q-learning -algoritmin arvot päivitetään (Comi 2018).

Watkins (1992) määrittelee Q-funktion arvioinnin ja välittömän palautteen pohjalta tapahtuvan jatkuvan arvojen päivittämisen pohjautuvan niin kutsuttuun **Bellmanin yhtälöön**, joka esittää tiettyyn tilaan kuuluvan maksimipalkkion vaatiman parhaan lähestymistavan löytyvän välittömän palkkion R(s, a) ja mahdollisen tulevaisuuden palkkion maxQ'(s', a') summana. Funktion sisältä löytyvä α eli oppimisparametri merkkaa niin kutsuttua **oppimistahtia** (engl. learning rate), jolla määritetään sen alueen laajuus jolta uusi Q:lle annettava arvio voi syrjäyttää vanhan arvion (Watkins 1992). γ eli **vähennyskerroin** (engl. discount factor) punnitsee jo saatujen palkkioiden ja tulevien palkkioiden tärkeyden; mitä lähempänä vähennyskertoimen arvo on numeroa 1, sitä suurempi painoarvo annetaan tuleville vahvistuksille (Watkins 1992).

Algoritmi käyttää hyväkseen bootstrappingia ylläpitämällä arvioita siitä, kuinka hyvä kulloinen tila-toiminto-pari Q(s, a) eli näiden Q sen mielestä on verrattuna kuinka hyvänä se pitää seuraavaa tilaa Q(s', a') (Yannakakis & Togelius 2018, s. 75). Lisäksi algoritmilla on Yannakisin ja Togeliuksen (2018, s. 75) mukaan käytössään yksiaskelinen täysivarmuuskopio (engl. one-step-depth, full-breadth backup), jonka avulla Q voidaan arvioida ottamalla huomioon kaikki sen arvot kaikista mahdollisista toiminnoista a' uudessa tilassa s'. Optimaalinen menettelytapa voidaan lopulta laskea funktion avulla saatujen Q-arvojen perusteella, kun opetettava agentti tilassa s valitsee toiminnon a jolla on korkein Q(s, a)-arvo (Yannakakis & Togelius 2018, s. 75).

Tiivistäen optimaalisten arvojen löytäminen vaatii siis bootstrappingia; jatkuvaa iterointia, "itsensä kehittämistä" ja erilaisten skenaarioiden kokeilemista, jotka kerta kerralta muuttavat Q-funktion arvoa lähemmäs optimaalia suoritusmallia ja maksimoivat varsinaista palkintosignaalia pitkän aikavälin tavoitteiden saavuttamiselle. Algoritmin suorittaminen lopetetaan yleensä, kun jokin tietty määrä iteraatioita on suoritettu tai haluttu **konvergenssin** laatu – eli tavoitteiden mukainen suoritus – on saavutettu (Yannakakis & Togelius 2018, s. 75).

Q-learning -algoritmilla on rajoitteita, joista suurimmaksi ongelmaksi nousee laajoissa tila-toiminto-avaruuksissa tapahtuva monimutkaista laskentaa vaativa hyvin

korkea resurssitarve (Yannakakis & Togelius 2018; Chen, C. 2011). Tila-toiminto-taulukon eli Q-taulukon koon kasvaessa laskennalliset tarpeet allokoitavalle muistille ja sieltä haettavalle informaatiolle kasvavat taulukon koosta riippuen hyvinkin suuriksi. Tämän lisäksi resurssitarpeesta riippumatta saattaa aiheutua hyvin pitkää konvergenssia – eli odotusaikaa halutun ratkaisun ja optimaalisen suoritusmallin sekä etenemispolun löytämiselle – aikavaativuuden ollessa eksponentiaalinen tila-toiminto-avaruuden kokoon. Rajoitteiden selvittäminen vaatii siis jonkinlaisen keinon pienentää kyseessä olevan tila-toiminto-avaruuden kokoa vaarantamatta optimaalisen lopputuloksen kannalta tärkeiden tila-toiminto-parien sivuuttamista. Parempaa suoritusaikaa on pyritty saavuttamaan käyttämällä Q-funktion arvojen arvioimiseen suoraan neuroverkkoja sivuuttamalla Q-taulukon asettamat rajoitukset ja antamalla tämän sijaan tiivistetyt representaatiot itse agentille. (Yannakakis & Togelius 2018, s. 75-76)

3 Tekoäly videopeleissä

Tutkielmassa on tähän mennessä käsitelty sen aiheen kannalta tärkeitä käsitteitä ja taustaa teoreettisemmin, mutta tässä kappaleessa syvennytään tekoälyn tarkasteluun tarkemmin videopelien ja videopelihahmojen kontekstissa. Ensin syvennytään NPC-tekoälyhahmojen tarkasteluun ja siihen kuinka ne yleensä videopeleissä toimivat. Tämän jälkeen painopiste siirretään koneoppimisen käyttöön videopeleissä ja lopuksi tarkastellaan erinäisin malliesimerkein videopelitekoälyissä esiintyvää oppimisen simulointia.

3.1 NPC-hahmot

Non-player character tai **non-playable character** (**NPC**), eli ei-pelaajahahmo on Yannakisin ja Togeliuksen (2018, s. 92) mukaan videopeleissä erilaisia rooleja saava sellainen ei-ihmispelaajan käsissä oleva hahmo, jota tietokoneen ohjelmakoodi ohjaa. Tietokoneohjattujen hahmojen tekoäly videopeleissä perustuu erilaisten sääntöjen ja ehtolauseiden pohjalta toteutettaviin päätöksentekoprosesseihin (Ponsen 2004), jotka on luotu erilaisten ympäristöjen ja pelaajan käyttäytymisen ympärille (Ganguly ym. 2018). Parhaimmillaan prosessit koostuvat moninaisista erilaisista systeemeistä ja algoritmeista, joita yhdistellään hienovaraisesti monipuolisen ja immersiivisen pelikokemuksen luomiseksi, kuten Alien NPC:n tekoälyä pelissä Alien Isolation käsittelevä alaluku 3.3.1 pyrkii osoittamaan.

Pelihahmot omaavat aina jonkin asteisen tekoälyn, mutta tämä vaihtelee pelikohtaisesti ja yksinkertaisimmillaan hahmon tekoäly voidaan nähdä pelkkään navigoimiseen keskittyvänä **reitinhakuälynä** (engl. path finder) kuten **A* -algoritmi**, joka keskittyy etsimään pelikartalta järkevän reitin paikasta A paikkaan B (Yannakakis & Togelius 2018, s. 41, 110). Tällaisia tekoälyn ohjastamia rajatuissa ympäristöissä toimivia ohjelmia, jotka osaavat itse päättää mitä niiden on tehtävä saavuttaakseen niille annetut tavoitteet, voidaan kutsua myös **agenteiksi** (Kellomäki 2012). Russell ja Norvig (2016, s. 46) määrittelevät agenttien koostuvan yleensä neljästä eri osasta:

syötteenluvusta eli sensoreista, toimintoja suorittavasta järjestelmästä eli aktuaattoreista, tietovarastosta eli muistista sekä tiedon käsittelyjärjestelmästä.

Agentit saavat syötteensä pelimaailmasta ja suorittavat näiden mukaisesti siellä erilaisia toimintoja. Agenttien tekemät toiminnot on rakennettu yleensä niin kutsuttujen **Ad-Hoc Behavior Authoring**-ryhmään kuuluvien perinteisten algoritmien, kuten tilasta tilaan johtavien **äärellisten automaattien** (engl. finite state machine), **päätös-** (engl. decision tree) ja **käytöspuiden** (engl. behavior tree) sekä **tavoitteeseen suuntaavan tehtävän suunnittelijan** (engl. goal oriented action planning, GOAP) avulla (Simpson 2014; Owens 2014). Käytännön kohteina mainittakoon historiallisen merkittävyytensä vuoksi Bungien Halo 2 (2004), joka käytännössä popularisoi videopeleihin käytöspuut sekä Sierra Entertaimentin FPS-kauhupeli F.E.A.R. (2005), johon GOAP-algoritmi alun perin suunniteltiin (Yannakakis & Togelius 2018, s. 11-12).

Vihollistekoälypelihahmojen suunnittelussa tärkeinä elementteinä on muun muassa niiden uskottavuus, toimintavarmuus ja peli-intoa yllä pitävä optimaalinen vastus (Sweetser & Wyeth 2005). Pelistä riippuen tekoälyvastusten pitää pystyä reagoimaan pelaajan tekoihin ja toimimaan vuorovaikutuksessa maailman ja pelaajan kanssa erilaisin tavoin (Yannakakis & Togelius 2018, s. 92-97). Vihollistekoälyhahmojen on hyvä pitää pelaaja ajan tasalla tavoitteistaan vaikkapa ilmoittamalla tämänhetkisestä tilastaan tai toimimalla tahallisesti hieman vastuuttomasti esimerkiksi ampumalla aina hetken aikaa ohitse, kun pelaaja nousee ja tähtää näitä suojasta. Vihollistekoälyhahmojen osumatarkkuus lähtee kuitenkin välittömästi nousuun pelaajan pysyessä pois suojasta. (Russell 2010)

Videopelitekoäly voi olla fiksu ja haastava, mutta sen pitää aina antaa pelaajan pystyä olemaan fiksumpi (Lidén 2003). Pelien tekoäly eroaa teknistieteellisistä tekoälyn käyttökohteista pyrkimällä optimaalisen ratkaisun ja toiminnan löytämisen sijaan luomaan halutunlaisen illuusion jostakin todellisen maailman osasta olemalla uskottava, muttei turhauttava ja siis peli-intoa ja iloa ylläpitävä (Kellomäki 2012). Lidén (2003) lisää teknistieteellisen tekoälyn käyttökohteisiin liittyen, että videopelien ensisijaisena tavoitteena on olla akateemisten ja teollisten sovellusten rinnalla

mahdollisimman viihdyttäviä teknisen toteutuksen sijaan.

Esimerkiksi Uncharted 4:n (Naughty Dog 2016) suunnittelijana toiminut Matthew Gallent totesi GamesIndustry.biz:lle antamassaan haastattelussa peli-ilon syntyvän liian realististen NPC-hahmojen ja kovan haasteen sijaan mielenkiintoisesta pelattavuudesta: *"We had to take a step back and say that the goal of these (patrolling guards) NPCs searching for the player isn't to find the player. It's to present interesting gameplay"* (Sinclair 2016).

3.2 Koneoppiminen videopeleissä

Michen (2015) ja Emighin (2016) mukaan koneoppimista on yritetty käyttää videopelitekoälyissä jonkin verran, mutta toistaiseksi melko huonolla menestyksellä. Syy koneoppimisen vähäiselle käytölle videopelitekoälyissä johtuu pääosin siitä, että esimerkiksi vahvistetun oppimisen kaltaisen virheistä oppimisen toteuttaminen videopeliin vaatii valtavasti muistia, laskentatehoa ja kehitysresursseja, joka ei aina välttämättä edes johtaisi halutunlaiseen lopputulokseen (Stephenson 2018; Miche 2015). Peleissä kaiken pitää myös toimia varmasti ja hallitusti juuri sen mukaisesti, kun ne on tarkoitettu toimimaan (Yannakakis & Togelius 2018, s. 171). Algoritmit eivät voi siis lähteä liikaa omille teilleen tai muutoin peli rikkoontuu ja pelikokemus luonnollisesti kärsii. Tästä syystä ohjelmoijat eivät ole pääasiallisesti olleet kovin luottavaisia koneoppimisen kaltaisiin epädeterministisiin tekniikoihin, eikä mullistavan tekoälyn kehittämiselle jää aikaa painopisteen ollessa vielä tällä hetkellä graafisella näyttävyydellä paremman tekoälyn sijaan (Mac Namee 2004). Yannakis (2012) perustelee tekoälyn hidasta kehitystä videopeleissä sillä, ettei akateemisella tutkimuksella ole toistaiseksi ollut tarjota merkittävästi parempia ja peleihin sopivasti skaalautuvia tekoälyllisiä ratkaisuita.

Ponsen (2004) perustelee koneoppimisen vähäistä käyttöä videopeleissä kertomalla kuinka pelifirmat pyrkivät varovaisuuteen koneoppimisen kanssa sen ollessa raskasta järjestelmälle, se voi oppia väärin ja sen muokkaaminen saamaan oikeanlainen tulos on vaikeaa. Ponsen (2004) kirjoittaa myös koneoppimisen mahdollistavan

paremman ja älykkäämmän tekoälyn, joka pelien kehityksen kannalta olisi tarpeen. Lisäksi useissa moderneissa videopeleissä NPC-hahmojen halutaan lopulta olevan tarpeeksi ennalta-arvattavia ja kontrolloitavia pelimäisyyden säilymiseksi (Stephenson 2018; Miche 2015; Yannakakis & Togelius 2018).

Vaikka koneoppimista ei edellä mainittujen seikkojen takia ole kovin paljoa videopelitekoälyissä käytetty, löytyy tähän poikkeuksia. Yhtenä esimerkkinä mainittakoon juuri vahvistettua oppimista sisuksissaan käyttävä Lion Head Studiosin Black & White (2001), jossa metodia käytetään pelihahmojen opettamiseen (Yannakakis & Togelius 2018, s. 12). Pelissä pelaaja on jumala, joka ohjaa kylän asukkeja ja valtavaa olentoa. Ohjattavan olennon tekoäly koostuu niin sanotusta uskomus-toive-aikomus -päätöksentekomallista (engl. belief-desire-intention decision model), jonka toteuttamiseen on käytetty päätöspuita ja yksinkertaisia **perseptroneja** (Yannakakis & Togelius 2018, s. 12). Perseptronit ovat käytännössä biologisia neuroneja jäljitteleviä yksinkertaisia malleja, joita käytetään **neuroverkkojen** (engl. artificial neural network, ANN) rakentamiseen (Millington & Funge 2009, s. 646-648). Black and White -pelissä olentojen toiveet tiettyjä tavoitteita kohtaan on mallinnettu juuri näiden yksinkertaisten perseptronien kautta (Yannakakis & Togelius 2018, s. 12). Olento muodostaa jokaista halua kohden uskomuksen, johon se on muodostanut voimakkaimman mielipiteen jotka siis ovat esitetty päätöspuissa. Vahvistetun oppimisen kautta olennon päätöksenteko muuttuu sen mukaan, miten pelaaja sitä palkitsee ja toruu ja se oppii toimimaan erilaisin tavoin (Evans 2002). Black & Whiten tekoälyjärjestelmä on edelleen yksi monimutkaisimmista videopelitekoälyistä (Millington & Funge 2009, s. 8).

Vahvistetun oppimisen kautta tapahtuvaa pelihahmon toimintaa ja oppimista voidaan lisäksi tarkastella Q-learning -algoritmin avulla Kuviossa 2 (Comi 2018) olevan, tila-toiminto-avaruutta havainnollistavan **Q-arvotaulukon** pohjalta. Esimerkkitaulukossa jonkin tekoälyagentin eli tietokoneohjatun pelihahmon jokaista mahdollista tilaa (engl. state) kohtaan on määrätty neljä erilaista toimintoa – kuviossa Right, Left, Up ja Down – joista Q-learning -algoritmin mukaisesti paras lähestymistapa eli korkeimpaan maksimaaliseen lopputulokseen ja palkkioon johtava toi-

minto on päivitetty vihreinä indikoituihin laatikoihin. (Comi 2018) Pelihahmo on esimerkiksi liikkunut tilassa 2 oikealle, joka on halutun lopputuleman, kuten pelihahmon nopean etenemisen kannalta ollut selvästi parempi ratkaisu kuin samassa tilassa vasemmalle, ylös tai alas liikkuminen.

State	Right	Left	Up	Down
1	0	0.31	0.12	0.87
2	0.98	-0.12	0.01	0.14
3	1	0.10	0.12	0.31
4	0.19	0.14	0.87	-0.12

Kuvio 2: Q-arvotaulukko, joka päivittää Q-funktion avulla tilakohtaisesti parhaan lähestymistavan eri toiminnallisuuksille (Comi 2018).

Sen lisäksi, että koneoppimista on käytetty osana pelitekoälyä, on neuroverkkoja ja vahvistettua oppimista yhdistelevää niin kutsuttua **syvää vahvistettua oppimista** (engl. deep reinforcement learning, DRL) käytetty useissa peleissä jälkikäteen opettelemaan niiden pelaamista (Mnih 2015). Algoritmi oppi esimerkiksi pelaamaan suosittua Snake-peliä sujuvasti 150 pelikerran, eli noin viiden minuutin pelaamisen jälkeen, vaikka se ei alussa osannut eikä tuntenut pelin sääntöjä (Comi 2018). Merkittävänä ja tulevaisuuden kannalta mielenkiintoisena tapauksena syvän vahvistetun oppimisen käytöstä videopeleissä on OpenAI:n (2019) viittä neuroverkkoa yhdistelevä **OpenAI Five**-tekoäly, joka pelaa päivittäin 180 vuoden edestä itseään vastaan elektronisen urheilun (engl. esports) rintamalla hyvin suosittua ja kompleksia Dota 2 -peliä (Valve Corporation 2013) kehittyen siinä jatkuvasti paremmaksi ja tehokkaammaksi (OpenAI 2019).

Huhtikuussa 2019 OpenAI Five oli onnistunut kehittymään pelissä sille tasolle että se voitti vuoden 2018 The International Dota 2 -suurturnauksen voittaneen OG-

mestaruusjoukkueen näytösottelussa 2-0 muutamin peliin tehtyjen rajoitusten avulla, esimerkiksi kaikki pelin sankarihahmot ja kyvyt eivät olleet käytössä (OpenAI 2019). Tekoäly oli tähän mennessä pelannut peliä itseään vastaan noin 45 000 vuoden edestä ja saavutusta voidaan rajoituksistaan huolimatta pitää tärkeänä edistysaskeleena OpenAI:n tavoittelemaa ja ihmiskuntaa hyödyntävää autonomista **yleistä tekoälyä** (engl. artificial general intelligence, AGI) kohtaan (OpenAI 2019).

3.3 Oppimisen simulointi pelitekoälyissä käytännössä

Tässä kappaleessa tarkastellaan alaluvussa 3.3.1 käytännönläheisesti tärkeimpiä elementtejä, jotka muodostavat Alien-pelihahmon tekoälyn ja simuloidun oppimisen videopeliin Alien: Isolation (The Creative Assembly 2014), joka myös inspiroi tämän tutkielman aiheen valinnassa. Lähteenä alaluvussa toimii pelin johtavana tekoälyohjelmoijana toimineen Andy Brayn konferenssiesitelmä *It's in the Vents: The AI of Alien Isolation* (2016) ja AI and Games -kanavan tämän pohjalta tekemä tiivistelmä *The Perfect Organism—The AI of Alien: Isolation* (2017). Huomautettakoon, että tavoitteena oli alun perin tutkia jotakin Alien-pelihahmon tekoälyjärjestelmän tietyistä osa-alueista tarkemmin, mutta vähäisen virallisen ja yksityiskohtaisen tiedon vuoksi järkevimmäksi ratkaisuksi osoittautui tarkastella kaikkia pelihahmon tekoälyn tärkeimpiä pääkohtia korkeammalla tasolla sen tekoälyjärjestelmän kokonaiskuvan saamiseksi. Alaluvussa 3.3.2 esitellään lisäksi muutama muu konkreettinen NPC-hahmojen oppimisen simuloinnin kannalta olennainen ja mielenkiintoinen esimerkkiteos.

3.3.1 Alien: Isolation

Alien-pelihahmon tekoäly Alien: Isolationissa on lyhyesti ja yksinkertaisesti kokoelma erilaisia **päätöksentekojärjestelmiä** (engl. decision making system), joiden avulla Alien pyrkii löytämään tiensä aina oikeaan aikaan ja paikkaan rikkomatta tarinan rytmitystä ja sen kulkua. Pelissä Alienia ei voi tappaa, vaan sitä voi satuttaa ja sen voi pelottaa pois, mutta Alienin saadessa pelaajan kiinni tappaa se tämän välittömästi, jonka ympärille tietyt osat mekaniikasta on rakennettu.

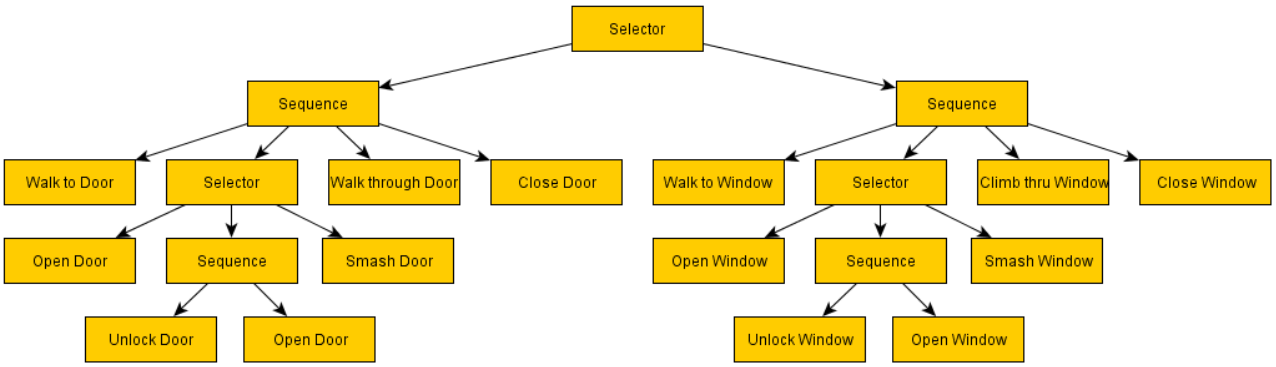
Ensimmäisenä Alienin tekoälyn (engl. Alien AI) kannalta tärkeänä järjestelmänä peliä ohjaa niin kutsuttu **kaksitasojärjestelmä** (engl. two tier system), jossa on kaksi pääasiallista Alienin käyttäytymistä ja pelimaailman tapahtumia ohjaavaa tekijää:

- **Macro AI** eli niin kutsuttu **AI Director** on kaikkitietävä säätelijä tai ohjaaja, joka hallitsee tilanteita ja Alienin **uhkatasoa** (engl. menace gauge). Ohjaaja mittaa esimerkiksi etäisyyttä pelaajaan, aikaa jonka Alien viettää pelaajan lähistöllä, aikaa jonka pelaaja viettää Alienin näkökentässä sekä aikaa, jonka Alien on näkyvillä pelissä olevassa liikkeentunnistinsensorissa ja tarpeeksi lähellä hyökkäämään. Alienin uhkatason kaltaista stressimittaria on käytetty myös Valven Left For Dead -pelisarjassa (Booth 2009). Lisäksi ohjaaja pitää jatkuvaa laskua sekä pelaajan, että Alienin tämänhetkisestä statuksesta.
- **Micro AI** on reaktiivinen ja "aistiensa" varassa toimiva varsinainen Alien-tekoäly. Tekoäly saa ohjeita Macro AI:lta ja reagoi sekä pelaajan tekemiin toimenpiteisiin että AI Directorin käskyihin, jotka yhdessä auttavat Alienia pelaajan etsimisessä ja hahmon navigoimisessa.

Toisena tekoälyjärjestelmänä Alienin Micro AI:sta löytyy **työjärjestelmäksi** (engl. job system) nimetty kokonaisuus, joka sanelee Alienille muun muassa mitä tehtäviä ja missä sijainnissa nämä tehtävät suoritetaan sekä jokaisen tehtävän tärkeysjärjestyksen. Saman kaltaista järjestelmää on aikaisemmin käytetty muun muassa Bioshock Infinitessä (Irrational Games 2013) ja siitä löytyvässä Elizabeth-pelihahmossa (Robertson 2014). Työjärjestelmä mahdollistaa Alienin tekoälylle kaksi mahdollista tilaa:

- **Aktiivinen**-tila ("Front Stage"), jossa Alien tutkii pelaajan tämänhetkistä olinaluetta tai tapahtumaa, jossa se esimerkiksi kuulee jonkin pelaajan aiheuttaman äänen.
- **Passiivinen**-tila ("Back Stage"), joka tulee voimaan kun uhkataso nousee tarpeeksi korkeaksi, jolloin Alien palaa esimerkiksi takaisin ilmastointiaukkoihin. Siellä Alien saattaa edetä esimerkiksi huoneeseen, jonne olettaa pelaajan siirtyvän seuraavaksi.

Kolmantena kokonaisuutena Alienin varsinainen käyttäytyminen perustuu erilaisiin **käytöspuihin** (engl. behavior tree). Yleisellä tasolla käytöspuut on haarautettu jakautuviin hierarkkisiin haaroihin, jotka eri tavoin ohjailevat tekoälyn käyttäytymistä ja päätöksentekoa. Jokainen puun haara päättyy aina lehtisolmuun, joissa kaikki varsinaiset tekoälyä ohjaavat komennot ja tapahtumat sijaitsevat. Kun uusi tapahtuma aktivoituu, lähtee tekoäly laskemaan käytöspuun juuresta ylöspäin kohti sopivinta lehtisolmua kyseisen tilanteen ratkaisemiseksi (Kuvio 3) (Simpson 2014). Itse pelissä näissä käytöspuissa on yli 100 solmua, joista 30 korkean tason (engl. top decision layer) solmua on vastuussa siitä millaista käytöstä Alien suorittaa. Nämä korkean tason solmut toimivat suurissa puun alitasoissa, jotka ovat puolestaan vastuussa tiettyihin tehtäviin liittyvistä erityisistä ali-käytösmalleista. Toisin sanoen alipuut (engl. sub trees) ja niissä sijaitsevat lehtisolmut on määrätty eri tehtäviin kuten hyökkäämiseen, etsimiseen ja niin edelleen.



Kuvio 3: Esimerkki käytöspuun toiminnasta (Simpson 2014).

Tietyt osat käytöspuusta ovat pelin alussa lukossa, mutta aukenevat pelin edetessä tiettyjen ehtojen täyttyessä. Tämä luo vaikutelman siitä, että Alien omaksuu uusia käytösmalleja ja oppii kokemuksesta sopeutumaan pelityyliisi, jolloin pelaaja pyritään pitämään jatkuvasti jännityksessä. Mikään näistä ehdoista ei vaadi sellaisia tapahtumia jotka johtaisivat pelaajan kuolemaan, koska tällöin tekoäly saattaisi kehittää epäreilun etulyöntiaseman. Tämän sijaan kaikki ehdot on piilotettu pelaajan itsensä pelityyliin ja toistuviin toimintamalleihin liitännäisesti. Tästä syystä Alienin "oppimistahti" voi vaihdella pelaajasta riippuen. Vaikka tiettyjen alipuiden käytös-

mallit ja ehdot eivät olisi täyttyneet pelaajan toimintojen ansiosta, niin tietyt käytöspuun osat aktivoituvat kaikesta huolimatta pelaajan edetessä tarpeeksi pitkälle erityisiin tallennuspisteisiin. Näin Alien pysyy dynaamisesti pelaajan vauhdissa kun pelaaja oppii pelaamaan peliä paremmin.

Neljäntenä ja samalla viimeisenä tärkeänä Alienin tekoälyjärjestelmänä ovat sen tekoälyyn piilotettu kokoelma erilaisia reitinhaku-käyttöisiä **sensoreita**, joilla Alien voi tarpeeksi lähellä pelaajaa ollessaan havaita erilaisia ääniä aina pelaajan askelista aseen ampumaääniin ja pelaajan hallussa olevan liiketunnistinlaitteen piippaukseen. Jokaisen sensorin kantama riippuu aina tapahtuvan äänen tyypistä.

Reitinhaku-sijaintityypit ovat:

- **Search-tila**, eli sub-optimaalinen jokaisen navigointipisteen tarkastelutila, jossa Alien liikkuu tiettyä paikkaa kohti ja etsii kaikki määrätyt sijainnit. Tarkastelu voi tapahtua missä tahansa satunnaisessa ja ennalta-arvaamattomassa järjestyksessä, joka johtaa joskus dynaamisesti tilanteisiin, joissa Alien palaa esimerkiksi takaisin paikkoihin, joita se on juuri hetki sitten tutkinut. Tarkastelun etenemiseen vaikuttavat myös erilaiset tilanteet, kuten jos Alien kuulee vaikkapa jonkin äänen tai huomaa pelaajan.
- **Spot-tila**, eli huomaamistila, jossa Alien pysähtyy paikoilleen ja katsoo sinne suuntaan missä jokin huomiota herättänyt tapahtuma tapahtuu.

Alienille on lisäksi määritelty muutamia sellaisia alueita, jonne se ei pysty menemään jottei pelissä synny liian epäreiluja ja turhauttavia tilanteita. Alien ei "teleporttaa" eikä ole "kaikkitietävä" käytännössä koskaan. Poikkeus tähän tapahtuu kahdesti pelin 12–18 tuntisen kampanjan aikana tilanteissa, joissa Alien siirretään suoraan toiselle alueelle juonellisista syistä tapahtuviin välianimaatioihin. Kaiken tässä alaluvussa käsitellyn ohella pelissä on lukuisia muita pienempiä sen toimintaa ohjaavia mekaniikkoja ja järjestelmiä, mutta pelin tärkeimmän elementin ja entiteetin, Alienin toiminta rakentuu pohjimmiltaan tässä alaluvussa käsiteltyjen järjestelmien ja metodien varaan.

3.3.2 Muita käytännön esimerkkejä

Alien: Isolation ja aiemmin mainitut teokset eivät ole ainoita oppimista simuloineita videopelejä, joissa videopelitekoäly opettelee toimimaan tietyillä tavoin sen mukaan kuinka pelaaja niitä pelaa, vaan joukosta löytyy useita muita esimerkkejä, joista tässä alaluvussa esitellään muutama.

Esimerkiksi Forza Motorsport -pelisarjassa (Turn 10 Studios 2005–) on niin kutsuttu **Drivatar AI**, joka kerää dataa pelaajan ja muiden pelaajien pelaamisesta ja pelityylistä sekä muovaa näiden perusteella uniikin tietokoneohjatun ajajan, jota vastaan pelaaja voi kilpailla tai jonka pelaaja voi vaihtoehtoisesti asettaa kilpailemaan muita pelaajia vastaan (Gandolfi 2017). Silent Hill: Shattered Memories (Climax Studios 2009) puolestaan kerää ja tallentaa pelaajan lukuisien valintojen ja pelaamisen perusteella erilaista informaatiota rakentaakseen tälle lopussa uniikin psykologisen profiilin (Guardiola & Natkin 2015).

Toisenlaisena esimerkkinä manuaalisen vaikeusasteen valinnan sijaan videopelien haasteen säätelyssä käytetään toisinaan **dynaamiseksi vaikeusasteen säätelyksi** (engl. dynamic difficulty adjustment, DDA) kutsuttua metodia, joka muuttaa automaattisesti pelin ominaisuuksia, käyttäytymistä ja skenaarioita reaaliajassa pelaajan kykyjen mukaan, jolloin pelaaja ei tylsisty pelin ollessa liian helppo tai turhaudu sen ollessa liian vaikea (Zohaib 2018). Dynaaminen vaikeusasteen säätely voidaan nähdä olennaisena osana NPC-hahmojen oppimisen simuloinnisssa, kuten seuraavat esimerkit osoittavat.

Esimerkiksi useissa ajopeleissä on käytössä niin kutsuttu **kuminauhatekoäly** (engl. rubber band AI), joka laittaa tietokoneen ohjaamat kuskit ajamaan sellaista vauhtia, että niitä vastaan ajava pelaaja tuntee jatkuvasti kilpailevansa muiden kanssa huolimatta siitä, kuinka hyvin tämä peliä pelaa (Hunicke 2005). Dynaaminen vaikeusasteen säätely on käytössä myös Valve Corporationin Half-Life 2:een (2004) kehitetyssä **Hamlet-järjestelmässä** (Hunicke 2005), jossa pelimaailmasta löytyvät ammuslaatikot, terveyspaketit ja muut resurssit ovat sitä harvemmassa, mitä menestyksekkäämmin pelaaja peliä pelaa. Vastaavasti pelaajan kuollessa usein, on pelin

DDA-tekoäly huomattavasti avokätisempi pelimaailmasta löytyvien tavaroiden ja resurssien kanssa (Hunicke 2005).

Siinä missä Half-Life 2:n dynaaminen vaikeusasteen säätely ei resurssien ohella vaikuta vihollishahmojen toimintaan, ohjaa Capcomin Resident Evil 4:ssä (2005) käytetty DDA-tekoäly puolestaan pelaajaa vastaan sitä enemmän ja sitä haastavampia vihollishahmoja, mitä paremmin pelaaja peliä pelaa. Hyvin pelissä etenevä pelaaja saa vastaansa suuria ryhmittymiä erilaisia agressiivisia vihollistyyppejä epävarmemman pelaajan taistellessa samoissa tilanteissa selvästi vähempilukuisia ja passiivisempia joukkoja vastaan (Brown 2015).

Vastaavasti Konamin Metal Gear Solid 5: Phantom Painissa (2015) vihollishahmot oppivat kehittämään erilaisia taistelutaktiikoita sen pohjalta millä tavoin pelaaja peliä pelaa. Jatkuvasti pääosumia tavoitteleva pelaaja huomaa pian saavansa vastaan enemmän kypäriin varustautuneita sotilaita, kun taas lähitaisteluita ja raakaa voimaa suosiva pelaaja tulee kohtaamaan tiellään parempiin suojuksiin varustautuneita haulikkomiehiä (Khan 2015). Pelaajan hävitessä jatkuvasti alkaa peli ehdottamaan pelaajalle pelikokemusta huomattavasti helpottavaa, mutta pelihahmon uskottavuuden kyseenalaistavaa ja suorituspisteet vähentävää kanahattua (Brown 2015).

4 Yhteenveto

Tässä tutkielmassa käsiteltiin tekoälyllistä oppimista videopelikontekstissa ei-pelaavien NPC-hahmojen tasolla. Työn tavoitteena oli tekoälyhahmojen toiminnan, näihin rakennettujen erilaisten tekoälyllisten ratkaisujen ja -oppimisen toteutuksen tarkasteleminen moderneissa videopeleissä erilaisten esimerkkien keinoin. Videopelitekoälyn tarkastelemisen apuna ja teoreettisena pohjana avattiin tekoälyyn liittyviä yleisiä käsitteitä sekä koneoppimista, johon syvennyttiin paremmin vahvistetun oppimisen ja siihen liittyvän Q-learning -algoritmin avulla.

Vaikka uskottavalla videopelitekoälyllä on vielä paljon kehityskohteita (Bourassa & Massey 2012, s. 40) ja tähän liitetyn koneoppimisen kanssa ollaan vielä melko varhaisessa vaiheessa, eikä varsinaisille koneoppimistekoälymalleille tällä hetkellä ole modernien videopelien ja pelityyppien joukossa niiden pelimäisen luonteensa ansiosta juuri tarvetta, omaavat jatkuvasti kehittyvät ja monimutkaistuvat videopelit valtavan potentiaalin erilaisten vaihtoehtoisten tekoälyratkaisujen toteuttamisessa (Yannakis 2012). Etenkin vahvistettu oppiminen, neuroverkot ja niiden kanssa suoritetut näytöt ja mallitapaukset videopelitekoälyn opettamisesta erilaisiin suorituksiin ovat osoittaneet koneoppimisen toimivan käytännössä jo nyt, vaikka käytössä olevien algoritmien monimutkaiset tavat käyttää ja käsitellä massiivista dataa halutun lopputuloksen saavuttamiseksi vaatii lisää tutkimusta (Schmidhuber 2015).

Koneoppimiselle löydetään silti jatkuvasti uusia käyttökohteita, jolloin dataa analysoidaan enemmän ja löydetään säännönmukaisuuksia, joiden ansiosta monimutkaisten algoritmien videopeleille elintärkeät seurattavuus ja toimintavarmuus kasvavat. Jatkuvan teknologisen kehityksen, tutkimuksen ja oman oppimisemme seurauksena pelaajan toimiin dynaamisesti reagoivat skriptaamattomat ja uniikit käytösmallit, uudet ja realistisemmat pelikokemukset sekä ihmispelaajan kykyihin vertautuvat tekoälyvastukset ovat vain ajan kysymys. Pelien tekoälyllinen kehitys on ollut melko hidasta audiovisuaaliseen puoleen verrattuna (Miche 2015), mutta edellä mainitut väistämättömät edistysaskeleet ja kasvava kiinnostus pelien käyttämisestä tekoälyn tutkimuskohteena mahdollistavat kuitenkin muutosta.

Yhtenä seuraavista potentiaalisista askelista tekoälyllisessä kehityksessä voidaan odottaa todennäköisyyksiin, vastustajan käyttäytymiseen ja muihin kerättävissä oleviin tietoihin perustuvien pokerin kaltaisten **epätäydellisen informaation** pelien tuomien kehityshaasteiden selvittämistä, kuten millä tavoin tietokone voitaisiin saada analysoimaan pelien informaatiota ja rakentamaan strategioita, jotka pystyisivät haastamaan ammattilaispelaajat. Tämän saavuttaminen olisi tekoälytieteessä tärkeä merkkipaalu ja edistäisi pelien ohella muun muassa erilaisten palvelu- ja hoitoalalla avustavien sovellusten kehittämisessä (Moravčík ym. 2017). Tapa miten käsitämme videopelit ja niiden pelaamisen nyt, voi olla jotain aivan muuta ja erilaista tulevien vuosikymmenten saatossa. Se jota nykyään kutsumme hiekkalaatikoksi voi tulevaisuudessa olla kokonainen leikkikenttä, tai jotain vielä suurempaa, jossa ainoana rajana toimii oma mielikuvituksemme.

Tieteelliset lähteet

Alpaydin, E. (2016). *Machine Learning : The New AI*. MIT Press, Cambridge.

Chen, C. (2011). Hybrid MDP based integrated hierarchical Q-learning. *Science China Information Sciences*, 54(11), 2279-2294.

Duc, L. M. (2008). Hierarchical Pathfinding and AI-Based Learning Approach in Strategy Game Design. *International Journal of Computer Games Technology*. Hindawi Publishing Corporation.

Emigh, M. S. (2016). Reinforcement Learning in Video Games Using Nearest Neighbor Interpolation and Metric Learning. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(1), 56-66.

Erdem, A. N. & Halici, U. (2016). Applying Computational Aesthetics to a Video Game Application Using Machine Learning. *IEEE Computer Graphics and Applications*, 36(4), 23-33.

Gandolfi, E. (2017). Gaming Mirrors At Play Through Ludic Data-selves. *Academicus : International Scientific Journal, MMXVII*, (16), 88-104.

Ganguly, R., Rithvik Reddy, D., Venkataraman, R., & S, S. (2018). Review on foreground artificial intelligence in games. *International Journal of Engineering & Technology*, 7(2.8), 453-455.

Guardiola, E. & Natkin, S. (2015). A Game Design Methodology for Generating a Psychological Profile of Players. *Serious Games Analytics* (s. 363-380). Springer.

Hunicke, R. (2005). *The Case for Dynamic Difficulty Adjustment in Games*, (265), 429-433.

Jain, A., Duin, R. & Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1), 4-37.

Kapitanova, K. & Sang, H.S. (2012). Machine Learning Basics. *Intelligent Sensor Networks: The Integration of Sensor Networks, Signal Processing and Machine Learning* (s. 3-29). CRC Press.

Kaplan, J. (2016). *Artificial Intelligence*. Oxford University Press.

Levesque, H. (2011). *The Winograd Schema Challenge*. University of Toronto.

Mac Namee, B. (2004). *Proactive Persistent Agents — Using Situational Intelligence to*

Create Support Characters in Character-Centric Computer Games (väitöskirja, University of Dublin, Trinity College).

Miche, Y. (2015). Meme representations for game agents. *World Wide Web*, 18(2), 215-234. Springer.

Millington, I. & Funge, J. (2009). *Artificial Intelligence for Games*. CRC Press.

Mnih, V. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529.

Mohammed, M., Khan, M.B. & Bashier, E.B.M. (2017). *Machine Learning: Algorithms and Applications* (s. 1-34). CRC Press.

Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., & Bowling, M. (2017). DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337), 508-513.

Ponsen, M. (2004). *Improving adaptive game AI with evolutionary learning* (pro gradu -tutkielma, Lehigh University).

Russell, S. & Norvig, P. (2016). *Artificial intelligence: A modern approach (Third edition, Global edition)*. Harlow: Pearson Education Limited.

Schaeffer, J., & Van den Herik, H. J. (2002). Games, computers, and artificial intelligence. *Artificial Intelligence*, 134(1-2), 1-7.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61, 85–117.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1), 1-47.

Searle, J. (1980). Minds, Brains, and Programs. *Behavioral and Brain Sciences*, 3(3), 417-457.

Sutton, Richard S, & Andrew G Barto. (2018). *Reinforcement learning: An introduction (Second Edition)*. MIT press.

Sweetser, P. & Wyeth, P. (2005). GameFlow: A Model for Evaluating Player Enjoyment in Games. *Computers in Entertainment (CIE)*, 3 (2005). The University of Queensland.

Szita, I. (2012). Reinforcement Learning in Games. *Reinforcement Learning* (s. 539-577). Springer.

Thagard, P. (2005). *Mind : Introduction to Cognitive Science (Second Edition)*. MIT Press.

Turing, A. (1950). Computing Machinery and Intelligence. *Mind*, 59, 433-460.

Watkins, C. (1992). Q -learning. *Machine Learning*, 8(3), 279-292.

Yampolskiy, R. & Fox, J. (2013). Safety Engineering for Artificial General Intelligence. *Topoi*, 32(2), 217-226.

Yannakakis, G. (2012). *Game AI revisited*. IT University of Copenhagen.

Yannakis, G. & Togelius, J. (2018). *Artificial Intelligence and Games*. Cham: Springer International Publishing.

Zohaib, M. (2018). Dynamic Difficulty Adjustment (DDA) in Computer Games: A Review. *Advances in Human-Computer Interaction*, (2018), 1-12.

Muut lähteet

Booth, M. (2009). *The AI Systems of Left 4 Dead*. Valve Corporation.

Bourassa, M., & Massey, L. (2012). *Artificial Intelligence in Games*. Technical Memorandum No. DRDC Ottawa TM 2012-084.

Bray, A. (2016). *It's in the Vents: The AI of Alien Isolation*. nucl.ai Conference 2016. Saatavilla WWW-muodossa <URL: http://events.nucl.ai/track/systemic/>. Viitattu 6.2.2019.

Brown, M. (2015). *What Capcom Didn't Tell You About Resident Evil 4*. Game Maker's Toolkit. Saatavilla WWW-muodossa <URL: https://www.youtube.com/watch?v=zFv6KAdQ5SE&feature=youtu.be>. Viitattu 28.4.2019.

Comi, M. (2018). *How to teach AI to play Games: Deep Reinforcement Learning*. Towards Data Science. Saatavilla WWW-muodossa <URL: https://towardsdatascience.com/how-to-teach-an-ai-to-play-games-deep-reinforcement-learning-28f9b920440a>. Viitattu 23.3.2019.

Evans, R. (2002). Varieties of Learning. *AI Game Programming Wisdom*. Charles River Media.

Kellomäki, T. (2012). *TIE-11300 Tietotekniikan vaihtuva-alainen (Peliohjelmointi)-kurssin luentomoniste. Luku 9: Tekoäly*. Turun yliopisto.

Khan, H. (2015). *MGSV: Phantom Pain Enemies Response System, Defense, Vehicles Guide*. SegmentNext. Saatavilla WWW-muodossa <URL: https://segmentnext.com/2015/09/05/mgsv-phantom-pain-enemies-response-system-defense-vehicles-guide/>. Viitattu 28.4.2019

Lidén, L. (2003). Artificial Stupidity: The Art of Intentional Makes. *AI Game Programming Wisdom 2* (s. 41–48). Charles River Media.

Neittaanmäki P., Lehto, M., Nyrhinen R., Ojalainen A., Pölönen I., Rautiainen I., Ruohonen T., Tuominen H., Vähäkainu P., Äyrämö S. & Äyrämö S-M. (2018). *Tekoälyn perusteita ja sovelluksia*. Jyväskylän yliopisto.

OpenAI. (2019). *OpenAI Five*. Saatavilla WWW-muodossa <URL: https://openai.com/five/>. Viitattu 25.4.2019.

Owens, B. (2014). *Goal Oriented Action Planning for a Smarter AI*. Envato Tuts+.

Saatavilla WWW-muodossa <URL: http://gamedevelopment.tutsplus.com/tutorials/goal-oriented-action-planning-for-a-smarter-ai--cms-20793>. Viitattu 20.3.2019.

Robertson, S. (2014). *Creating BioShock Infinite's Elizabeth*. Game Developers Conference 14, San Francisco, California. Saatavilla WWW-muodossa <URL: https://www.gdcvault.com/play/1020545/Creating-BioShock-Infinite-s>. Viitattu 28.4.2019.

Russell, B. (2010). *The Secrets Of Enemy AI In Uncharted 2*. Gamasutra. Saatavilla WWW-muodossa <URL: http://www.gamasutra.com/view/feature/134566/the_secrets_of_enemy_ai_in_.php>. Viitattu 28.4.2019.

Simpson, C. (2014). *Behavior trees for AI: How they work*. Gamasutra. Saatavilla WWW-muodossa <URL: http://www.gamasutra.com/blogs/ChrisSimpson/20140717/221339/Behavior_trees_for_AI_How_they_work.php>. Viitattu 20.3.2019.

Sinclair, B. (2016). *Uncharted 4 is not as scripted as you might think*. GamesIndustry.biz. Saatavilla WWW-muodossa <URL: https://www.gamesindustry.biz/articles/2016-11-04-uncharted-4-not-as-scripted-as-you-might-think>. Viitattu 10.3.2019.

Stephenson, J. (2018). *6 Ways Machine Learning will be used in Game Development*. Logikk. Saatavilla WWW-muodossa <URL: https://www.logikk.com/articles/machine-learning-in-game-development>. Viitattu 23.3.2019.

Thompson, T. (2017). *The Perfect Organism—The AI of Alien: Isolation*. AI and Games. Saatavilla WWW-muodossa <URL: https://becominghuman.ai/the-perfect-organism-d350c05d8960>. Viitattu 6.2.2019.