**Emil Keränen**

# Vertaileva tutkimus koneoppimisen hyödyntämisestä videopelien reitinhaussa

Tietotekniikan pro gradu -tutkielma

20. huhtikuuta 2022

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

**Tekijä:** Emil Keränen

**Yhteystiedot:** emil.a.keranen@student.jyu.fi

**Ohjaaja:** Tommi Kärkkäinen

**Työn nimi:** Vertaileva tutkimus koneoppimisen hyödyntämisestä videopelien reitinhaussa

**Title in English:** Comparative study of utilizing machine learning in video games' pathfinding

**Työ:** Pro gradu -tutkielma

**Opintosuunta:** Tietotekniikka

**Sivumäärä:** 18+0

**Tiivistelmä:** TODO: tiivistelmä suomeksi

**Avainsanat:** koneoppiminen, videopeli, reitinhaku, syvä vahvistusoppiminen

**Abstract:** TODO: In english

**Keywords:** machine learning, video game, pathfinding, deep reinforcement learning, Soft Actor Critic, Machine Learning Agents, Unity

# Sisällys

1 JOHDANTO ..... 1

2 REITINHAKU VIDEOPELEISSÄ ..... 3
2.1 Pelialueen esitystavat ..... 3
2.2 Reitinhakualgoritmit ..... 3
2.2.1 A*-algoritmi ..... 4
2.2.2 A*-algoritmin variaatiot ..... 4
2.3 Reitinhaun ongelmat ..... 5

3 UNITY ..... 6
3.1 Machine Learning Agents ..... 6

4 KONEOPPIMINEN ..... 7

5 TUTKIMUSSTRATEGIA/METODI JA SEN VALINTAPERUSTEET ..... 8

6 AINEISTON KERUUN SUUNNITTELU ..... 9

7 AINEISTON KERUU ..... 10

8 AINEISTON ANALYYSI ..... 11

9 TULOKSET ..... 12

10 JOHTOPÄÄTÖKSET ..... 13

LÄHTEET ..... 14

11 LIITTEET ..... 15

# 1 Johdanto

- Tutkimuksen kohteena koneoppimisen tehostama reitinhaku videopeleissä ja sen vertaaminen heuristiseen A*-algoritmiin.

- Yleisesti reitinhaulla tarkoitetaan alku- ja loppupisteen välisen reitin selvittämistä. Useimmiten tarkoituksena on löytää lyhin reitti väistellen samalla matkan varrella olevia esteitä.

- Reitinhakua tarvitaan videopelien lisäksi myös mm. robotiikassa.

- Videopeleissä reitinhaku ilmenee pääasiassa tekoälyagenttien suorittamana toimintana, joten tässä tutkimuksessa keskitytään agentteihin. Näitä agentteja kutsutaan myös ei-pelaaja-hahmoiksi (engl. non-player-character, NPC).

- Ei-pelaaja-hahmot ja itseasiassa videopelien reitinhaku vertautuvat hyvin robotiikkaan ja robottien reitinhakuun.

- Sekä robotiikassa että videopeleissä toiminta-alue voi muuttua hyvinkin paljon reaaliajassa, jolloin reitinhaun täytyy sopeutua muutoksiin nopeasti. Käytetyt ratkaisut sen sijaan voivat vaihdella näiden kahden osa-alueen välillä: robotiikassa tarkkuus ja turvallisuus nousevat tärkeimmiksi ominaisuuksiksi ja vastaavasti videopeleissä nopeus määrittää reitinhaun "hyvyyden".

- Reitinhaku on aina ollut vaativa ongelma videopeleissä, mutta nykyään reitinhaun ongelmallisuus voidaan useimmissa tapauksissa sivuuttaa laatimalla heuristinen ratkaisu A*-algoritmin avulla.

- Koneoppiminen mahdollistaa aiemman kokemuksen hyödyntämisen myöhemmässä toiminnassa. Agentteja voidaan kouluttaa harjoitteludatan avulla, jolloin ne oppivat toimimaan tuntemattomissa tilanteissa. Koneoppimisen ansiosta reitinhaku-agentti voidaan opettaa toimimaan vaativissa ja dynaamisissa pelialueissa, joissa muuttuvat esteet ja alueen labyrinttimäisyys heikentävät A*-algoritmin toimintaa.

- Tutkimuksen ideana on käyttää Unity-pelimoottorille luotuja koneoppimisagentteja (engl. Unity Machine Learning Agents) ja opettaa niitä erilaisten pelialueiden avulla. Opettamisen

jälkeen agentteja testataan oikeilla pelialueilla ja verrataan tuloksia A*-algoritmilla saatuihin tuloksiin.

- ML-agents perustuu PyTorch-kirjastoon ja mahdollistaa vahvistusoppimisen hyödyntämisen.

- Tensorboard-lisäosan avulla voidaan visualisoida palkkioiden keskiarvot ja opetuksen edistyminen opetuksen aikana.

- Pelialueiden on tarkoitus olla monimutkaisia ja dynaamisia, koska A*-algoritmi suoriutuu yksinkertaisista reitinhakutehtävistä moitteettomasti.

# 2 Reitinhaku videopeleissä

Reitinhaulla tarkoitetaan yksinkertaisimmillaan reitin selvittämistä kahden pisteen välillä. Se on yksi videopelien tekoälyn ja myös robotiikan tunnetuimmista ja haastavimmista ongelmista, jota on tutkittu jo muutaman vuosikymmenen ajan (Cui ja Shi 2011; Abd Algfoor, Sunar ja Kolivand 2015). Reitinhakua esiintyy monissa eri peligenreissä, kuten roolipeleissä ja reaaliaikaisissa strategiapeleissä, joissa ei-pelaaja-hahmoja (non-player-character, NPC) määrätään liikkumaan ennaltamäärättyyn tai pelaajan määräämään sijaintiin väistellen samalla vastaantulevia esteitä (Cui ja Shi 2011).

## 2.1 Pelialueen esitystavat

Yleisin alueen esittämistapa on ruudukko (engl. grid), joka sisältää yksittäisiä ruutuja (engl. tile). Ruudut merkitään ennalta vapaiksi tai esteiksi. Vapaista ruuduista muodostetaan verkko, jolloin jokainen vapaa ruutu vastaa yhtä verkon solmuista. Vierekkäisiä solmuja yhdistävät linkit, joita pitkin reitinhaku tapahtuu. Solmujen vierekkäisyys voi tarkoittaa horisontaalista ja vertikaalista vierekkäisyyttä (neljä suuntaa) tai näiden lisäksi myös diagonaalista vierekkäisyyttä (kahdeksan suuntaa). (Botea ym. 2013; Abd Algfoor, Sunar ja Kolivand 2015).

(Kuva ruudukosta / verkosta?)

Vähemmän käytettyjä alueen esitystapoja ovat kolmiointi ja kuusikulmiointi (oikea termi kuusikulmioinnista?) (Abd Algfoor, Sunar ja Kolivand 2015). Alueen kolmiointi ja siihen perustuvat TA*- ja TRA*-algoritmi osoittautuivat moninkertaisesti nopeammiksi suurissa pelialueissa verrattuna A*-algoritmiin (Demyen ja Buro 2006). Kuusikulmioihin perustuvat alueet ja reitinhakualgoritmit ovat tuottaneet lupaavia tuloksia muun muassa robotiikan tutkimuksessa (Abd Algfoor, Sunar ja Kolivand 2015).

## 2.2 Reitinhakualgoritmit

Graafin lyhyimmän polun ongelmaa on tutkittu jo vuosikymmenten ajan. Vanhimmat ja tunnetuimmat algoritmit, Dijkstran algoritmi (Dijkstra ym. 1959) ja A*-algoritmi (Hart, Nilsson ja Raphael 1968), esiteltiin jo 50- ja 60-luvuilla. A*-algoritmi on selvästi tunnetumpi videopelien ja robottien reitinhaussa nopeutensa ansiosta (Cui ja Shi 2011; Abd Algfoor, Sunar ja Kolivand 2015; Botea ym. 2013). Tässä tutkimuksessa keskitytään tarkemmin A*-algoritmiin ja sen variaatioihin.

### 2.2.1 A*-algoritmi

A*-algoritmi on hyvin tunnettu paras-ensin -reitinhakualgoritmi (termi kuntoon?), joka hyödyntää heuristista arviointifunktiota lyhimmän reitin etsimiseen (Cui ja Shi 2011; Duchoň ym. 2014). Algoritmin toiminta tapahtuu seuraavasti: jokainen aloitussolmun vierekkäinen solmu arvioidaan kaavan f(n)=h(n)+g(n) mukaisesti, jossa n on solmu, h(n) on heuristinen etäisyys solmusta n maalisolmuun ja g(n) on todellinen etäisyys aloitussolmusta solmuun n. Näistä solmuista matalimman f(n)-arvon solmu käsitellään seuraavaksi, jolloin kyseisen solmun vierekkäisten solmujen f(n)-arvot lasketaan. Tämä prosessi jatkuu, kunnes maalisolmu saavutetaan. (oikeat matemaattiset merkinnät). Heuristiikan ollessa nolla A*-algoritmista tulee Dijkstran algoritmi.

A*-algoritmilla on kolme esitettyä ominaisuutta (Hart, Nilsson ja Raphael 1968). A*-algoritmi löytää reitin, jos sellainen on olemassa. Reitti on optimaalinen, jos heuristiikka on luvallinen(admissible?) eli arvioitu etäisyys on lyhyempi tai yhtä suuri kuin todellinen etäisyys. Lopuksi mikään muu algoritmi samalla heuristiikalla ei käy läpi vähemmän solmuja kuin A*-algoritmi eli A* käyttää heuristiikkaa tehokkaimmalla mahdollisella tavalla. (Hart, Nilsson ja Raphael 1968; Cui ja Shi 2011). Luvallisia heuristiikkoja ovat solmujen vierekkäisyydestä riippuen Euklidinen etäisyys, Manhattan-etäisyys, Chebyshev-etäisyys tai Octile-etäisyys (Duchoň ym. 2014; Botea ym. 2013). Manhattan-etäisyyttä käytetään pääasiassa neljän suunnan ja Octile-etäisyyttä kahdeksan suunnan vierekkäisyyksissä (Botea ym. 2013).

### 2.2.2 A*-algoritmin variaatiot

A*-algoritmin heikkous on huono skaalautuvuus suurien alueiden reitinhakuun. Esimerkiksi kaksiulotteisessa ja esteettömässä 1000x1000 ruudukkoalueessa solmuja on miljoona, jolloin A*-algoritmin muistinkäyttö ja nopeus koituvat ongelmaksi. (Cui ja Shi 2011; Duchoň ym. 2014) Tämä ongelma esiintyy videopelien lisäksi etenkin robotiikassa, jonka seurauksena A*-algoritmi on inspiroinut tutkijoita monien eri variaatioiden kehittämiseen, kuten Theta*-, Phi*- ja HPA*-algoritmit (Duchoň ym. 2014).

## 2.3 Reitinhaun ongelmat

Reitinhaun on oltava videopeleissä nopeaa ja laskennallisesti tehokasta. Lisäksi reittien on näytettävä realistiselta pelaajalle. Yhden agentin staattisen ruudukkoalueen reitinhakuongelma on ratkaistavissa nopeasti ja tehokkaasti heuristisella A*-algoritmilla, mutta nykyään videopeleissä reitinhakuongelmat ovat monimutkaisempia. Reitinhakuongelmat pitävät sisällään esimerkiksi useamman agentin samanaikaista reitinhakua ja reaaliajassa muuttuvien alueiden reitinhakua. Nykytutkimus keskittyykin pääasiassa monimutkaisiin reitinhakuongelmiin.

Useamman agentin samanaikaisessa reitinhaussa alueella on useampi kuin yksi agentti ja jokaisella niistä on oma aloitus- ja lopetuspisteensä. Jos jokaisen agentin reitinhakuun sovelletaan A*-algoritmia, on vaadittu laskennallinen teho eksponentiaalinen agenttien lukumäärän suhteen O(b potenssiin k) (tähän merkintätapa ja lähde). Laskennallisen tehon lisäksi jokaisen agentin täytyy tarvittaessa väistää toisia agentteja ja mahdollisesti muita esteitä. A*-algoritmi osoittautuu riittämättömäksi ongelman ratkaisuun.

# 3 Unity

Unity on Unity Technologiesin kehittämä pelinkehitysalusta, joka sisältää oman renderöinti- ja fysiikkamoottorin sekä Unity Editor -nimisen graafisen käyttöliittymän (Juliani ym. 2018). Unityllä on mahdollista kehittää perinteisten 3D- ja 2D-pelien lisäksi myös esimerkiksi VR-pelejä tietokoneille, mobiililaitteille ja pelikonsoleille. Unitystä onkin vuosien mittaan tullut yksi tunnetuimmista pelinkehitysalustoista, jonka parissa työskentelee kuukausittain jopa 1.5 miljoonaa aktiivista käyttäjää ("Unity" 2022).

Viime vuosina Unityä on käytetty simulointialustana tekoälytutkimuksen parissa. Unity mahdollistaa lähes mielivaltaisten tilanteiden ja ympäristöjen simuloinnin 2D ruudukkokartoista monimutkaisiin pulmanratkaisutehtäviin, joka on sen suurimpia vahvuuksia simulointialustana. Kehitystyö ja prototypointi ovat Unityllä myös erityisen nopeaa. (Juliani ym. 2018).

## 3.1 Machine Learning Agents

Machine Learning Agents on Unitylle kehitetty koneoppimispaketti, jonka avulla peliin voidaan ottaa käyttöön koneoppimisagentteja.

# 4 Koneoppiminen

Soft Actor-Critic on Haarnojan ym. kehittämä syvä vahvistusoppimis algoritmi (Haarnoja ym. 2018).

# 5 Tutkimusstrategia/metodi ja sen valintaperusteet

- Empiirinen vertaileva tutkimus?

- Luodaan pelialueita, toteutetaan heuristinen A*-algoritmi, opetetaan koneoppimisagentit syvän vahvistetun oppimisen avulla (Soft Actor Critic -algoritmi) ja sijoitetaan koneoppimisagentit pelialueille. Tämän jälkeen ratkaistaan reitinhakutehtävä erikseen molemmilla menetelmillä ja verrataan saatuja tuloksia keskenään (mm. lasketaan tehtävään kulunut aika, suoriutuiko tehtävästä tietyssä ajassa vai ei).

# 6 Aineiston keruun suunnittelu

- Agentit koneopetetaan käyttäen Unity ML-agents -pakettia. Opetusprosessista saatu mallitiedosto (model, .onnx-tiedosto) liitetään agentin komponentiksi, jolloin agentti voi toimia pelialueella itsenäisesti.

- Tensorboardin avulla oppimisprosessia voidaan visualisoida esimerkiksi palkkioiden suhteen.

- Aineisto kerätään peliagenteilta jokaisen reitinhakuongelman aikana. Vähintään aika ja tieto siitä onnistuiko agentti tai A*-algoritmi otetaan talteen. Myös Tensorboardin muodostamia kuvaajia voidaan käyttää apuna koneoppimista arvioidessa (mm. palkkioiden keskiarvo opetuksessa).

- Ajan laskemiseen käytetään Unityn valmiita kirjastoja.

# 7 Aineiston keruu

- Reitinhakuongelmia on jokaista erilaista pelialuetta kohden yksi. Pelialueita luodaan ainakin muutama kymmen.

- Malli-tiedostoja (model, .onnx-tiedosto) luodaan muutamia eri parametreilla, jolloin voidaan verrata parametrien vaikutusta agentin suoriutumiseen.

- Ajan laskeminen alkaa, kun agentti käsketään siirtymään pisteestä A pisteeseen B. Ajan laskeminen loppuu, kun agentti saapuu pisteeseen B. Jos agentti ei syystä tai toisesta pysty ratkaisemaan ongelmaa eli ei saavuta pistettä B (esim. aikamaksimi saavutetaan, alueesta riippuen 20+ sekuntia), merkitään ratkaisun tilaksi "epätosi"ja jätetään aika-arvo tyhjäksi. Jos ratkaisu löytyy, merkitään ratkaisun tilaksi "tosi"ja asetetaan aika-arvoksi mitattu aika. Sama prosessi toistetaan A*-algoritmilla.

- Jos koneoppimisagentteja opetetaan eri määrällä dataa, ilmoitetaan myös opetusdatan määrä.

- Kaikki data kerätään yhteen tiedostoon ja käsitellään jälkikäteen.

# 8 Aineiston analyysi

- Kuvaajien avulla visualisoidaan kuluneita aikoja ja vertaillaan saatujen aikojen erotuksia.

- TODO: lisätietoja analyysista

# 9 Tulokset

- TODO

# 10 Johtopäätökset

- TODO

# Lähteet


Abd Algfoor, Zeyad, Mohd Shahrizal Sunar ja Hoshang Kolivand. 2015. “A comprehensive study on pathfinding techniques for robotics and video games”. *International Journal of Computer Games Technology* 2015.

Botea, Adi, Bruno Bouzy, Michael Buro, Christian Bauckhage ja Dana Nau. 2013. “Pathfinding in games”. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Cui, Xiao, ja Hao Shi. 2011. “A*-based pathfinding in modern computer games”. *International Journal of Computer Science and Network Security* 11 (1): 125–130.

Demyen, Douglas, ja Michael Buro. 2006. “Efficient triangulation-based pathfinding”. Teoksessa *Aaai,* 6:942–947.

Dijkstra, Edsger W, ym. 1959. “A note on two problems in connexion with graphs”. *Numerische mathematik* 1 (1): 269–271.

Duchoň, František, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico ja Ladislav Jurišica. 2014. “Path planning with modified a star algorithm for a mobile robot”. *Procedia Engineering* 96:59–69.

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel ja Sergey Levine. 2018. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. Teoksessa *International conference on machine learning,* 1861–1870. PMLR.

Hart, Peter E, Nils J Nilsson ja Bertram Raphael. 1968. “A formal basis for the heuristic determination of minimum cost paths”. *IEEE transactions on Systems Science and Cybernetics* 4 (2): 100–107.

Juliani, Arthur, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar ym. 2018. “Unity: A general platform for intelligent agents”. *arXiv preprint arXiv:1809.02627.*

“Unity”. 2022. Viitattu 4. huhtikuuta 2022. https://unity.com/.

- Panov, A., Yakovlev, K. ja Suvorov, R., Grid Path Planning with Deep Reinforcement Learning: Preliminary Results, Procedia Computer Science Volume 123, Pages 347-353, 2018, https://doi.org/10.1016/j.procs.2018.01.054

- X. Lei, Z. Zhang ja P. Dong, Dynamic Path Planning of Unknown Environment Based on Deep Reinforcement Learning, 2018, https://doi.org/10.1155/2018/5781591

- TODO: oikea merkintätapa ja muita lähteitä.

# 11 Liitteet

- Kuvat tai mallinnokset pelialueesta.

- Tensorboardin kuvaajat mm. agentin palkkioiden kehityksestä.