Tampereen yliopisto

Eero Tarri

# PELIMOOTTORIEN JA DIGITAALISTEN KAKSOSTEN HYÖDYNTÄMINEN TUOTANNONOHJAUKSESSA

Informaatioteknologia ja viestintä
Kandidaatintyö
Tarkastaja: Henri Lunnikivi
Joulukuu 2022

# TIIVISTELMÄ

Eero Tarri: Pelimoottorien ja digitaalisten kaksosten hyödyntäminen tuotannonohjauksessa
Kandidaatintyö
Tampereen yliopisto
Tietotekniikan tutkinto-ohjelma
Joulukuu 2022

---

Digitaaliset kaksoset tarjoavat tavan viedä raskaat ja hankalasti säädettävät laitteet helposti muokattavaan digitaaliseen ympäristöön. Digitaalisia kaksosia voidaan hyödyntää luomalla mahdollisimman tarkka virtuaalinen malli ympäristölle ja laitteille ja fyysisen maailman sijaan tehdä suunnittelu ja alustavat testaukset siellä nopeutetusti. Tehokas tapa luoda virtuaalisia ympäristöjä ja todellisen oloisia digitaalisia kaksosia on tehdä ne pelimoottoreilla. Tämän tutkimuksen tarkoituksena on selvittää, millaisia eri ominaisuuksia digitaalisten kaksosten toteuttamiseen vaaditaan ja löytyykö niitä tunnetuimmilta pelimoottoreilta.

Tässä kirjallisuuskatsauksessa määritämme digitaaliset kaksoset ja vertailemme kolmea merkittävintä pelimoottoria. Kirjallisuuskatsauksesta voidaan huomata, että ei ole täysin itsestään selvää, mikä pelimoottori on paras juuri digitaalisten kaksosten rakentamiseen. Unity3D ja Unreal Engine ovat tällä hetkellä markkinoilla saatavilla olevia pelimoottoreita. Usein jompikumpi näistä kahdesta on oikea valinta lähteä rakentamaan digitaalisia kaksosia hyötykäyttöön. Muun pelimoottorin valinta on järkevää, jos tietää tarkalleen, mitä ominaisuuksia pelimoottorilta kaipaa ja ne löytyvät jostain toisesta pelimoottorista.

Tutkimuksen perusteella valituilta pelimoottoreilta pitäisi löytyä riittävästi ominaisuuksia. Unity3D:llä ja Unreal Enginellä vakuuttavia ominaisuuksia oli kuitenkin enemmän kuin kolmantena tutkimukseen valitulla O3DE-pelimoottorilla. Erityisesti Unity3D vaikutti erittäin varmalta vaihtoehdolta digitaalisten kaksosten luomiseen helposti. Unreal Engine on menestyksekäs graafistesti hienoimpien tuotteiden tekemisessä. O3DE on riskialttein valinta, sillä sen kyvystä tehdä digitaalisia kaksosia on hyvin vähän näyttöä.

Avainsanat: digitaalinen kaksonen, pelimoottori, tuotannonsuunnittelu

# SISÄLLYSLUETTELO

# 1. JOHDANTO

Digitaalinen kaksonen (engl. *Digital Twin*, DT) näyttää samalta ja toimii funktionaalisesti samalla tavalla kuin se fyysinen laite, josta se on luotu. Ihmisen on hankala käsitellä teksti- tai numeromuotoista raakaa dataa ja kuvaajien ja taulukoidenkin käyttö näyttää vain osan todellisuudesta. Tähän avuksi voidaan ottaa käyttöön digitaaliset kaksoset, joilla voidaan toteuttaa halutut simuloinnit käyttäjälle luontevammalla tavalla eli visualisoimalla. Digitaalisia kaksosia voidaan luoda tehokkailla pelimoottoreilla, joiden tarkoitus on tarjota valmiita ominaisuuksia DT:n luomiseen. Aihe on juuri tällä hetkellä mielenkiintoinen, sillä nykyajan pelimoottoreilla on mahdollista tehdä helposti monenlaisia 3D-ympäristöjä ja komponentteja, joita jopa kaupallisesti saatavilla olevat laitteet voivat pyörittää. Näin voidaan tehdä pelin kaltaisia ohjelmistoja, joiden avulla tuotannon simulointi ja optimointi on helppoa, intuitiivista sekä nopeaa. Tämän tutkimuksen tarkoituksena on selvittää merkittävien pelimoottorien soveltuvuus digitaalisten kaksosten toteuttamiseen sekä vertailla niiden eroja.

Tuotannon toteuttaminen koostuu karkeasti kahdesta suuremmasta osasta, jotka ovat suunnittelu ja itse tuotanto. Kolmannessa teollisessa vallankumouksessa 1900-loppupuolella opittiin hyödyntämään fossiilisia polttoaineita ja automaattisia teollisia koneita. Tällä vastattiin tuotannon toteuttamiseen automatisoinnilla, kun työvoima korvattiin monen työntekijän kuormaan kykenevillä koneilla. Tämä pienensi tuotantoon kuluvaa aikaa ja kustannuksia ja teki suunnittelusta ainoan manuaalisen ja hankalan osan. Neljännessä teollisessa vallankumouksessa laitteiden lähettämää dataa ja niiden välisiä yhteyksiä hyödynnetään rakentamalla tietoteknisiä järjestelmiä, joilla suunnittelu voidaan automatisoida (Kritzler et al. 2017) samalla tavalla kuin polttoainetta hyödyntävillä tehdaslaitteilla automatisoitiin tuotannon manuaalinen työ. Sillä on tarkoitus korvata hitaammat tavat kuten arvokarttojen käsin laatiminen (Uhlemann et al. 2017) ja lokitietojen selaaminen vuorovaikutteisemmalla ja intuitiivisemmalla tai jopa automaattisella tavalla. Lähestyttävä tapa seurata ja vaikuttaa tuotantoon vapauttaa resursseja siihen erikoistuneilta ammattilaisilta ja helpottaa ulkopuolisten liittymistä suunnitteluun. Myös uusi käyttäjä oppii nopeammin järjestelmän käyttämisen, kun käyttöliittymä on ennestään tuttu, kuten suora 3D-mallinnettu kopio tehtaasta, jossa on jo pitkään työskennellyt.

Internet of Things (IoT) eli asioiden internet tarkoittaa teknologiaa, jossa laitteita yhdistetään kommunikoimaan keskenään verkon välityksellä. IoT-teknologia mahdollistaa laitteiden jatkuvan kommunikoinnin keskenään sekä datankerääjien kanssa. Laitteiden jakaman datan määrä on kasvanut ja kasvaa yhä. Datan määrä on kasvanut niin paljon, ettei ihminen pysty enää käsittelemään sitä muokkaamatta saatua dataa luettavampaan muotoon. IoT-teknologialla voidaan ottaa käyttöön reaaliaikainen simulointi. Digitaalisten kaksosten fyysiset ja virtuaaliset objektit toimiessaan IoT-laitteina eli ollessaan kytkettynä verkkoon luovat perustan reaaliaikaiselle simuloinnille. Ilman tätä ominaisuutta, data on siirrettävä jotakin muuta kautta, joka on lähes väistämättä hidasta ja manuaalista.

Pelimoottoreiden kykyä toteuttaa digitaalisia kaksosia ei mainosteta paljoa, eikä siitä löydä helposti tietoa dokumentaatiosta. Työn tavoitteena on selvittää kirjallisuudesta, mitä pelimoottorilta vaaditaan, jotta digitaalinen kaksonen voidaan toteuttaa mahdollisimman tehokkaasti ja helposti. Työssä perehdytään siihen, mitä ominaisuuksia kaksosilla on ja selvitetään, päästäänkö pelimoottorilla toivottuun lopputulokseen.

Toisessa luvussa kerrotaan tutkimukseen käytetyistä menetelmistä. Kolmannessa luvussa perehdytään digitaalisen kaksosen ominaisuuksiin ja menetelmiin sekä pelimoottoreiden ominaisuuksiin ja rajoituksiin. Neljännessä luvussa vertaillaan kolmen eri pelimoottorin ominaisuuksia. Viidennessä luvussa kootaan havainnot yhteen ja tehdään johtopäätökset pelimoottoreiden eduista.

# 2. TUTKIMUSMENETELMÄ

Työ tehtiin kirjallisuuskatsauksena eli siinä tutkittiin aikaisempia tutkimuksia ja kerättiin niistä olennaista aiheeseen liittyvää tietoa. Työssä käytettiin myös pelimoottorin arkkitehtuurista kertovaa kirjaa sekä pelimoottorien tuottajien tarjoamaa dokumentaatiota.

Aikaisempina tutkimuksina käytettiin tietokannoista ACM ja ScienceDirect löytyviä vertaisarvioituja artikkeleita sekä konferenssipapereita. Myös IEEE:n julkaisemia artikkeleita käytettiin erityisesti pelimoottorien vertailussa. Haussa käytettiin avainsanoja ”game engine” sekä ”digital twin” tai ”virtual twin”. Näiden lisäksi apuna käytettiin tuotantoon liittyvää sanastoa, kuten ”smart manufacturing”, ”production planning” ja

”industry 4.0”. Enimmäkseen hakusanoja käytettiin yksin tai niitä yhdistettiin AND-operaattorilla tarkempien tulosten saavuttamiseksi.

Osa olennaisesta informaatiosta löytyi heikosti tutkimuksista, joten oli haettava tietoa dokumentaatiosta. Tällaista tietoa olivat lähinnä yhteensopivuuteen liittyvät asiat, kuten tuontimuoto ja digitaalisen kaksosen pelimoottorikohtaiset viitekehykset. Pelimoottorit ovat jo jonkin verran tutkittuja, joten niistä pystyttiin keräämään tietoa kirjasta.

# 3. DT:N JA PELIMOOTTORIN TOIMINTA

Ensimmäisessä luvussa käsitellään digitaalisen kaksosen määrittelyä, luomisprosessia sekä käyttökohteita. Niiden lisäksi siinä tarkastellaan DT-teknologian tuomia lupauksia. Toisessa kappaleessa käsitellään pelimoottorien ominaisuuksia, rajoitteita ja reunaehtoja.

## 3.1 Digitaalinen kaksonen

Jones et al. (2020) kuvaavat digitaalisen kaksosen koostuvan kolmesta osasta. Osina ovat fyysinen olio, virtuaalinen olio sekä yhteydet niiden välillä. Fyysisellä oliolla tarkoitetaan jotakin konkreettista asiaa, kuten tehtaassa käytettävää konetta. Se on aina olemassa, vaikka sitä ei olisikaan kytketty virtuaaliseen olioon. Virtuaalinen olio on fyysisestä oliosta luotu mahdollisimman tarkka virtuaalinen kuvaus, joka toimii virtuaalisessa ympäristössä samalla tavalla kuin fyysinen olio toimii todellisessa ympäristössä. Oliolla tarkoitetaan tässä osaa, jota ei ole vielä linkitetty fyysisen tai virtuaalisen parinsa kanssa. Virtuaalisesta objektista tulee siis virtuaalinen kaksonen sitten, kun se on liitetty fyysiseen vastinpariinsa.

Digitaalisen kaksosen on tarkoitus luoda visuaalinen, skaalautuva ja helppokäyttöinen rajapinta suunnittelun ja suunnittelijan välille ja mahdollistaa kevyempi tapa testata lopputuloksia eri parametreilla. Digitaalinen kaksonen on erittäin hyödyllinen juuri sen vuoksi, että sitä voi siirtää ympäristössään lähes energiaa kuluttamatta, kun taas todellisessa ympäristössä siirto vaatisi huomattavasti enemmän aikaa ja muita resursseja.

DT-teknologia voi linkittää yrityksen simulointiin parantaen mallin tarkkuutta ja IoT:n ehkäisevää huoltoa sekä analyysiä varten (Hyren et al. 2022). DT-teknologiaan

siirtyminen vaatii investoinnin, jotta järjestelmä voidaan luoda ja sitä käyttävät henkilöt voidaan kouluttaa järjestelmän käyttämiseen, mutta muiden investointien tapaan hyöty tulee myöhemmin. Investointi on osittain taloudellinen, mutta suurimmaksi osaksi se tarkoittaa muutosta tuotantotavoissa. DT-teknologian avulla voidaan kerätä lähes mitä tahansa dataa ja hyödyntää se halutulla tavalla. Esimerkiksi DT:n ja 3D-ympäristön fyysisiä mittoja voidaan hyödyntää optimaalisen järjestyksen suunnittelussa interaktiivisesti tai laitteen sisäänmenoparametreja simuloimalla voidaan löytää optimaalinen suhde sisäänmenoille ja ulostuloille. Pelimoottorien hyödyntäminen digitaalisten kaksosten kanssa mahdollistaa skaalattavan ja tehokkaan toteutuksen.

DT-teknologiaa on tutkittu 2010-luvusta alkaen paljon. Ollessaan niin lyhyen aikaa tutkittu asia ei DT:lle ole vielä vakiintunut yleisesti hyväksyttyä tapaa rakentaa niitä tietyillä työkaluilla tai käyttää tietyllä tavalla. Aiheesta on kuitenkin tarjolla ehdotuksia viitekehyksille sekä demonstraatioita, kuinka digitaalisia kaksosia on onnistuneesti hyödynnetty sovelluskohteen simuloinnissa ja optimoinnissa.

Digitaalisen kaksosen etuna aikaisempiin toimintatapoihin on suurempi läpinäkyvyys. Läpinäkyvyys parantaa käytettävyyttä niin, että ammattilaisten sekä aloittelijoiden kyky analysoida tuotannon syöttämää dataa helpottuu ja markkinoille viemiseen kuluva aika lyhenee merkittävästi. Simulointi mahdollistaa tarkemman ennustamisen, jolloin tulevat huoltotoimenpiteet voidaan hoitaa ennakoiden ja virhetilanteisiin kuluvaa aikaa voidaan lyhentää ja kuluja välttää.

## 3.2 Digitaalisen kaksosen luominen

Adamenkon et al. (2020) mukaan digitaalisia kaksosia voi rakentaa kahdella tavalla: dataan perustuvalla tavalla sekä järjestelmän ominaisuuksiin perustuvalla tavalla. Datapohjaisessa mallissa tärkeintä on, että malli pääsee tarvitsemaansa dataan käsiksi. Tämä voi tapahtua esimerkiksi hyödyntämällä sensoridataa. Datapohjaisen digitaalisen kaksosen tarkoitus on muokata raaka monitorointidata oleelliseksi informaatioksi (Alejandro et al. 2020). Dataan pohjautuvan digitaalisen kaksosen etu on se, ettei kaikkea teknistä tietoa tarvita sen rakentamiseen. Malli luodaan kolmessa vaiheessa. Ensin kerätään tarvittava data. Toisena luodaan mallille olennaiset toiminnallisuudet. Toiminnallisuus saattaa tarkoittaa esimerkiksi sisään menevistä materiaaleista koostuvan sensoridatan vastaamaa ulostuloa tai sitä, kuinka laitteen käyttö vaikuttaa epäkuntoon menemiseen. Kolmannessa vaiheessa mallista tehdään mahdollisimman

käyttäjäystävällinen esimerkiksi visualisoimalla mallin luomat analyysit, jotta toivottu hyöty saadaan käyttöön mahdollisimman tehokkaasti.

Datapohjaiseen malliin verrattuna järjestelmäpohjaisen mallin luominen on huomattavasti hankalampaa. Pelkästään sensoridata ei riitä, vaan tarvitaan kaikki tekninen tieto, mitä on saatavilla. Tämä sisältää laitteen fyysiset mitat, materiaalin tai muun vastaavan informaation, joka on sovelluskohteen kannalta olennaista. Esimerkiksi fyysiset mitat ovat olennaisia, jos tarkoituksena on suunnitella tuotantolinjan sijoittelua, mutta paino ei välttämättä ole olennaista tietoa. Järjestelmään perustuva tieto on tärkeä tallentaa käyttökohteissa, joissa ei seurata pelkästään laitteen sisäänmenon ja ulostulon suhdetta, vaan esimerkiksi optimoidaan laitteiden sijoittelua tai suurinta mahdollista määrää tehtaassa.

Digitaalisen kaksosen voi luoda myös yhdistämällä dataan sekä järjestelmään pohjautuvat rakentamismetodit. Lopputuloksena on kaksonen, jolla voi tehdä sekä optimointia että simulointia ja näiden lisäksi myös tuotannonseurantaa.

### 3.2.1 Digitaalisten kaksosten tasot tuotannossa

Digitaalinen kaksonen ei käsitteenä kuvaa mitään tiettyä tasoa, vaan sillä voidaan kuvata eri kokoisia järjestelmiä. Qi et al. (2018) määrittelevät digitaalisen kaksosen niin, että se voi olla yksikkötason, systeemitason tai System of Systems -tason (SoS) digitaalinen kaksonen. Taon et al. (2022) mukaan yksikkötasoista digitaalista kaksosta voidaan verrata yksittäiseen laitteeseen, systeemitasoa tuotantolinjaan ja SoS-tasoa koko tehtaaseen.

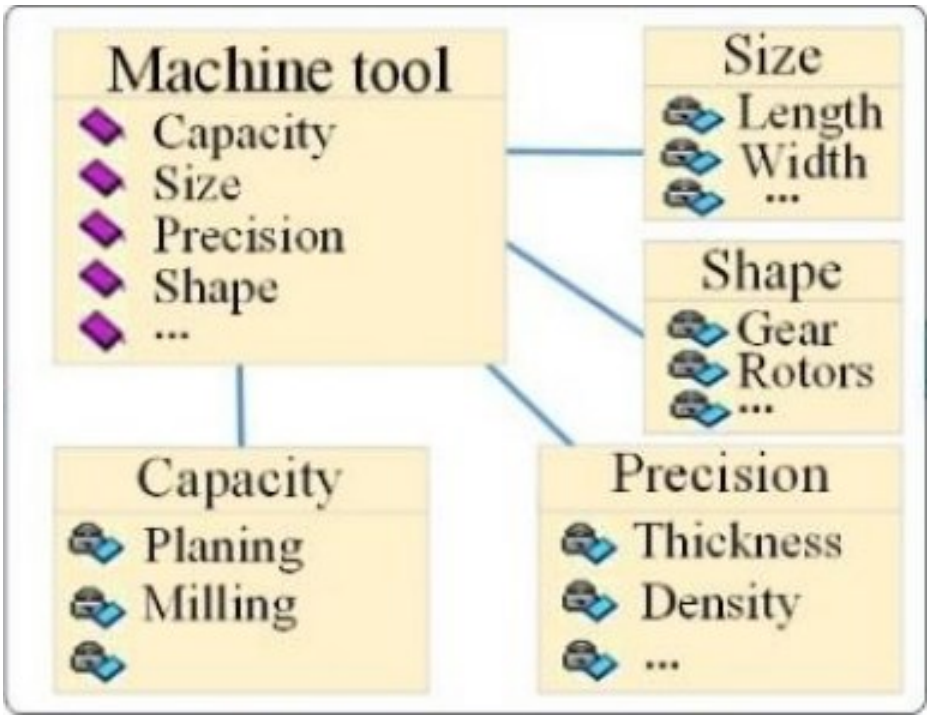
Yksikkötasoinen DT mahdollistaa toimintoja kuten seuranta, virheiden ennakointi ja huolto (Tao et al. 2022). Se voi koostua pienemmistä mallinnetuista osista tai olla yksi kiinteä malli. Tämän kaltaisen DT:n määrittävä tekijä on kuitenkin usein se, että se suorittaa yhtä funktiota. Yksikkötasoinen DT ei usein toimi enää toivotulla tavalla, jos siitä poistaa osia.

Systeemitason ja SoS-tason DT mallinnetaan edustamaan sen koostavien komponenttien välisiä suhteita ja ennustamaan koko järjestelmän lopputulosta (Tao et al. 2022). Systeemitason DT on järjestelmä, joka sisältää vähintään kaksi yksikkötason digitaalista kaksosta. Se ei sisällä muita järjestelmiä, joten lopputulos on yksinkertainen. Kun esimerkiksi yksikkötason komponentti tekee tuotteelle yhden vaiheen, systeemitason komponentti ottaa kaikki tuotteen materiaalit ja tuottaa valmiin tuotteen. SoS-tason digitaalisella kaksosella tarkastellaan kokonaisuutta. Se ei eroa systeemitasosta muuten kuin siten, että se sisältää pienempiä järjestelmiä. SoS-tason digitaaliset kaksoset ovat

erittäin tärkeitä tuotannonsuunnittelussa, sillä niitä hyödyntämällä voidaan ennakoida koko tehtaan tuottamaa tulosta ja muokata suunnitelmaa tehokkaammin ja dynaamisemmin kuin manuaalisesti kartoittamalla.

### 3.2.2 Digitaalisen kaksosen ominaisuudet tehdaslaitteessa

Niin fyysisellä laitteella kuin sen digitaalisella kaksosellakin on käyttökohteen mukaan määrittäviä ominaisuuksia. Usein olennaiset ominaisuudet voidaan nähdä suoraan siitä datasta, mitä laitteesta kerätään, mutta osa ominaisuuksista on mahdollisesti tallennettava erikseen. Ominaisuudet voidaan jakaa tarkoituksen perusteella moneen eri kategoriaan. Tärkeimpiä esimerkkejä näistä ovat perusominaisuudet, Quality of Service (QoS), kapasiteetti, status sekä sisäänmenot ja ulostulot (Qi et al. 2018).




***Kuva 1.*** *Mallipohja laitteen ominaisuuksista (Qi et al. 2018)*

Perusominaisuuksia ovat laitteen metatiedot, kuten nimi, yksilöivä ID ja sijainti. Näitä käytetään laitteen tunnistamiseen ja löytämiseen. QoS kuvastaa laitteen laitteen suoriutumiskykyä. QoS-ominaisuuksia ovat esimerkiksi hinta, aika, luotettavuus sekä kyky toimittaa haluttuja tuotteita. Kapasiteetilla tarkoitetaan asioita, joita laitteella ja kaksosella voidaan tehdä. Niihin liittyy laitteen tarkkuus, koko sekä prosessit. Statuksella seurataan laitteen tilaa, kuten ollaanko tyhjäkäynnillä, ylitäytetty tai esimerkiksi onko laite

huollossa. Näiden lisäksi ovat sisäänmenot eli materiaalit tai parametrit, joita laite vastaanottaa sekä ulostulot eli, mitä hyötytuotteita tai palveluita laite tuottaa.

### 3.2.3 Digitaalisen kaksosen käyttökohteet

Digitaalisia kaksosia voidaan hyödyntää monella eri tavalla. Dehghanimohammadabadi et al. (2021) kertovat DT:n soveltuvan ainakin simulointiin sekä optimointiin. Qi et al. (2018) ehdottavat niiden käyttämistä myös kouluttamiseen. Käyttötapa riippuu käyttäjän tarpeista, mutta se vaikuttaa digitaalisen kaksosen suunnitteluun. Yksinkertaisin tapa hyödyntää digitaalista kaksosta on visualisointi. Visualisoinnin tehtävä on tuoda fyysinen ympäristö virtuaaliseksi, jolloin ympäristön kokonaiskuvan hahmottaminen on mahdollista muutenkin kuin fyysisen olion näköetäisyydellä. Laitetta saattaa olla tarve tarkastella liikkeessä, mutta ennen kaikkea kokonaiskuvan tarkastelu helpottuu. Digitaalisen kaksosen voi loitontaa niin, että koko järjestelmä on nähtävissä menemättä fyysisesti korkealle. Laitteen voi purkaa osiin vaivatta sekä ilman hajoamisen riskiä. Visualisointia tarjoava DT vaatii todennäköisesti osittaisen järjestelmään perustuvan digitaalisen kaksosen rakentamisen.

Toinen vaihtoehto on simulointi. Simulointi on jo hieman haastavampaa kuin pelkkä visualisointi, sillä visualisointi ei vaadi samanlaista toiminnallisuutta kuin simulointi. Simulointiin ei riitä olion fyysisten ominaisuuksien esittäminen, vaan sille on luotava toiminnallisuutta liikkeen ja laitteiden läpiviennin esittämiseksi. On määriteltävä, mistä tilasta kaksonen lähtee ja mihin se päätyy. Dataan pohjautuvalla tavalla voidaan luoda tällaisia kaksosia. Tällöin olennaista on kaksosen sisäänmeno, ulostulo ja niiden välinen funktio. Käyttäjä saa tietoonsa vain ulostulon, mutta sitä on edelleen tutkittava vain datana. Simulointia voi olla myös visuaalinen simulointi, jolla nähdään, miten komponentit toimivat keskenään ja mikä on niiden lopputulos. Tähän tarvitaan paremmin määritelty järjestelmään perustuva DT. Myös olion siirtäminen on tavallaan simulointia, sillä lopputulos on aloituspisteen ja siirtymän funktio.

Optimoinnin voi toteuttaa samalla tavalla kuin simuloinnin, sekä datapohjaisena että järjestelmäpohjaisena. Esimerkki datapohjaisesta simuloinnista olisi algoritmipalvelulle luotu kaksonen, jota käyttäjä voisi käyttää algoritmin parametrien optimointiin. Kaksonen toteuttaa sen ja esimerkiksi tarkastelee, montako laitetta on järkevää hankkia säilyttäen hankinnan kustannustehokkaana. Järjestelmäpohjaisessa optimoinnissa on kyse ennemminkin datapohjaisen optimoinnin ja visualisoinnin yhdistelmästä.

Visualisointia ja simulointia yhdistämällä on mahdollista luoda kokonaisuus, josta on hyötyä myös esimerkiksi demonstroinnissa ja koulutuksessa. Uusille käyttäjille

ymmärrettäväksi ja helposti jaettavaksi luotu virtuaaliympäristö auttaa hahmottamaan kokonaiskuvan ja opettaa välineiden käyttöä vähintään yhtä helposti kuin esittelemällä niitä fyysisesti.

## 3.3 Pelimoottori

Kirjassaan Gregory (2018, kpl 1.3) toteaa, että termi ”pelimoottori” pitäisi todennäköisesti varata ohjelmistolle, joka on laajennettava ja sitä voi käyttää perustana monille erilaisille peleille tekemättä suuria muutoksia. Pelimoottori on termi, jota alettiin käyttää 1990-luvun puolivälissä, kun pelien komponentteja alettiin erottelemaan ja uudelleenkäyttämään. Tuottajien ydinkomponentit kierrätettiin uudempiin peleihin, joihin tehtiin vain uudet varat (engl. *assets*) eli säännöt, maailmat, estetiikat ja muut pelin yksilöivät ominaisuudet. Nykyäänkin pelimoottoreita käytetään siihen, ettei jokaista peliä tarvitse rakentaa tyhjästä, vaan toistuvat komponentit, kuten fysiikkamoottorit sekä efektit voidaan tuoda jo olemassa olevasta pelimoottorista.

Vieläkään ei ole niin hyvää yleistilanteeseen sopivaa pelimoottoria, että sillä voitaisiin toteuttaa minkälainen tahansa sovellus tehokkaammin kuin millään muulla markkinoilla olevalla vastineella. Jokainen moottori on optimaalisempi eri tilanteessa kuin toinen. Voidaan huomata, että mitä yleiskäyttöisempi moottori on kyseessä, sitä vähemmän optimaalinen se on toteuttamissaan tilanteissa (Gregory, J., 2018). Tästä syystä niillä tehtyjä pelejä on yleensä syytä muokata ja tehdä lisäyksiä jopa itse pelimoottoriin. Se ei kuitenkaan tarkoita, ettei epäoptimaalista tuotetta voisi valita kevyen ja yksinkertaisen sovelluksen rakentamiseen. Tällöin kyseessä on todennäköisesti yleiskäyttöisempi pelimoottori, jolla voi tehdä paljon erilaisia kevyitä ratkaisuja.

### 3.3.1 Tarvittavat ominaisuudet

Tärkeimmät ominaisuudet, mitä pelimoottorilta vaaditaan, on kyky vastaanottaa sille tarjottuja mallitiedostoja sekä toteuttaa tarvittavat toiminnallisuudet. Ilman näitä ominaisuuksia DT:n luominen ei ole mahdollista tai se on liian haastavaa ollakseen tehokasta. Pelimoottori toimii usein yhdessä 3D-työkalujen kanssa, joilla on eri vaatimuksia. Sovelluskohteen mukaan sen on oltava teknisiltä ominaisuuksiltaan sopiva. Suurissa sovelluksissa tärkeää voi olla tehokas ison kartan renderöinti, kun taas yhden robottikäden luomiseen tärkeämpää saattaa olla fysiikkamoottorin ominaisuudet ja kyky hallinnoida monimutkaisia liikeratoja. Jatkuva fysiikka on tärkeää erityisesti simuloidessa, jotta saadaan tarkka kuvaus siitä, miten tilanne etenisi fyysisessä ympäristössä.

Vaatimukset ovat käyttökohteesta riippuvaisia, mutta tietty perustaso on säilytettävä skaalattavuuden vuoksi.

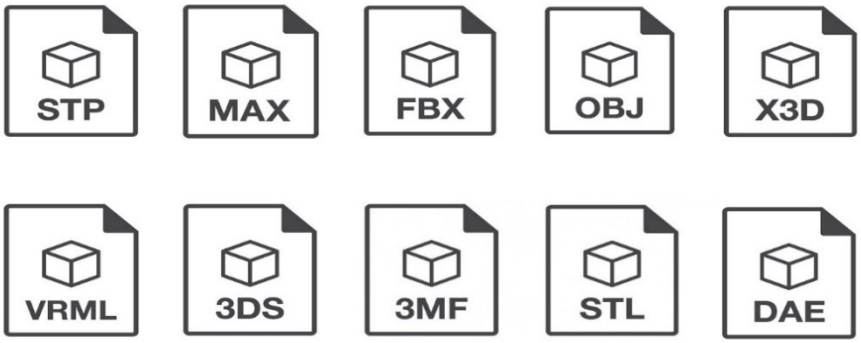
Tuotantoympäristössä valintaan vaikuttaa helppokäyttöisyys, lisenssiehdot sekä ajoympäristöt, sillä koko digitaalisten kaksosten ja pelimoottorien konseptilla haetaan tehokkuutta tuotantoon. Työkalujen käyttämisen hinta ei voi ylittää saatua hyötyä ja ennalta käytetyt ympäristöt saattavat vaikuttaa, jos pelimoottorin ajoympäristöt ovat rajalliset. Helppokäyttöisyys voidaan ajatella osana käyttöön kuluvia resursseja, sillä helppo työkalu vähentää kouluttamisen kuluja ja lyhentää käyttöön kuluvaa aikaa huomattavasti.

Tutkielmaan on valittu vertailtavaksi kolme tunnettua pelimoottoria. Unity Technologiesin Unity3D ja Epic Gamesin Unreal Engine on valittu niiden markkinoiden johtoasemansa vuoksi ja Amazonin Open 3D Engine (O3DE) on valittu avoimen lähdekoodin vertailukohteeksi.

### 3.3.2 Rajoitteet ja reunaehdot

Pelimoottoreita käyttäessä on hyvä huomioida mahdollisia haittatekijöitä jo suunnitteluvaiheessa. Nämä perustuvat käytännössä haluttuun lopputulokseen, joten on valittava sellainen järjestelmä, jolla siihen voidaan päästä.

Digitaalisten kaksosten visuaalista mallia ei välttämättä toteuteta pelimoottorilla, vaan jollakin muulla 3D piirtotyökalulla, kuten computer assisted designilla (CAD) tai skannaamalla. Ei voida taata käytetyn työkalun ulostulon olevan yhteensopiva pelimoottorin hyväksymien tiedostotyyppien kanssa (Zarco et al. 2021). Esimerkiksi SolidWorks 3D-mallinnustyökalun natiivi vientityyppi on .stl, mutta useimmat pelimoottorit haluavat hyödyntää .fbx tai .obj tiedostotyyppejä. Kehittäjälle jää kaksi vaihtoehtoa: valita yhteensopivat työkalut tai käyttää kolmannen osapuolen ohjelmistoa, jolla epäsopiva tiedostotyyppi voidaan muokata tarvittavaan muotoon. Lähtökohtaisesti on järkevää valita yhteensopivat työkalut, sillä tämä vähentää prosessiaskeleita ja kuluja.




***Kuva 2.*** *Yleiset 3D-mallien tiedostotyypit (CGIFURNITURE n.d.)*

Pelimoottorit hyödyntävät skriptausta, jotta toiminnallisuudet ja algoritmit saadaan yhdistettyä 3D-malleihin (Zarco et al. 2021). Skriptauskieli vaihtelee pelimoottorin mukaan ja useimmat hyödyntävät vain noin yhdestä kolmeen erilaista kieltä. Skriptikieli on tärkeää ottaa huomioon, sillä kehittäjien ohjelmointikielitaidoilla on vaikutus mallintamisen tehokkuuteen ja helppouteen. Skriptauskielen valinta ei ole välttämättä kuitenkaan suurin määrittävä tekijä pelimoottorin valinnassa, sillä usein skriptauskielet sisältävät valmiiksi komponentteja, joilla voidaan integroida muiden kielien koodia. Tämä voi kuitenkin tuottaa hankaluuksia ja lisätä kehityksen työmäärää. Uuden kielen opetteleminen ei välttämättä ole yhtä vaikeaa kuin vaikeasti opittavan pelimoottorin.

# 4. VERTAILU

Useimmat pelimoottorit eroavat toisistaan edes jollain tavalla. Pelimoottorin käytössä on otettava nämä erot huomioon, jos halutaan optimaalinen lopputulos tai tehdä samalla moottorilla useita erilaisia tuotteita.

## 4.1 Digitaalisten kaksosten vaatimat ominaisuudet

Käytännössä useimmat pelimoottorit pystyvät tekemään 3D-mallin, jolla on jotakin ohjelmoitua toiminnallisuutta. Tällöin voitaisiin jo puhua jonkinlaisesta digitaalisesta kaksosesta, mutta tuotantoa helpottaviin tarpeisiin ei olla vielä vastattu eli fyysisestä laitteesta voidaan tehdä helposti 3D-malli, mutta sillä ei vielä yksinään voi automatisoida tuotannonsuunnittelua. Tehokkuutta, julkaisua, kustannuksia ja lopputulosta ajatellen moni muukin asia vaikuttaa hyvään tulokseen, kuin pelimoottorin kyky luoda staattinen 3d-malli. Käyttöönottoon ja tuottamiseen kuluvaan aikaan vaikuttaa erityisesti käytetty skriptauskieli ja valmiit viitekehykset. Käyttövalmiin työkalun hankkiminen nopeuttaa prosessia ja vähentää kuluvia resursseja, sillä uusien käytäntöjen luominen on aikaa vievää ja haastavaa. Pelimoottorin fysiikkaominaisuudet ovat myös tärkeitä, kun halutaan realistinen simulaatio.

| | **Unity3D** | **Unreal Engine** | **O3DE** |
|---|---|---|---|
| **Korkean tason skriptaus** | C#, JavaScript | C++, Blueprints | C++ |
| **Fysiikkamoottori** | PhysX, Box2D, Unity Physics ja Havok | Chaos Physics | PhysX |
| **Jatkuva fysiikka** | Jäykät kappaleet, törmäys, nivelet, geometriakyselyt | Jäykät kappaleet, törmäys, kehittynyt vaatetus | Jäykät kappaleet, törmäys, nivelet, geometriakyselyt |
| **Lisenssiehdot** | Kiinteä maksu | Rojalti | Avoin lähdekoodi |
| **Ajoympäristöt** | PC, Web, Mobile, Xbox, PS | PC, Mobile, Xbox, PS | PC, Xbox, Mobile, PS, Web |
| **Valmiit viitekehykset** | Industrial Collection, AirSim (beta) | AirSim | N/A |

Taulukko 1. *Tärkeimmät digitaalisen kaksosen luomiseen liittyvät ominaisuudet*

Skriptauskieli saattaa vaikuttaa valintaan sen mukaan, mikä on kehittäjille vahvempi kieli. Unityssä käyttää C#- tai JavaScript-kieliä, mikä on tuttua kehittäjille, jotka ovat käyttäneet Windowsin .NET viitekehystä. Unreal Engine ja Amazon O3DE integroivat skriptauksen C++ -ohjelmointikielellä. Joskus halutaan kuitenkin käyttää resursseja toisesta ohjelmointikielestä. Tämän ongelman Unity3D ja Unreal Engine ovat ratkaisseet sisällyttämällä ristiin kääntämisen, joka kuitenkin saattaa kasvattaa simuloidun ohjelman viivettä ja laskea sen suorituskykyä. Ohjelmointikielillä C# ja JavaScript skriptaus on helpotettu abstraktimmalla standardikoodilla, joka kuvaa valmiin koodin toiminnallisuutta paremmin kuin matalamman tason kielillä kuten C++. Toisaalta C++-ohjelmointikielellä on korkeampi kontrolli tietokoneen arkkitehtuurista ja usein se tuottaa tehokkaamman lopputuloksen. Blueprints on Unreal Enginen oma graafisella käyttöliittymällä varustettu visuaalinen skriptauskieli.

Unity3D:n ja O3DE:n käyttämä PhysX-fysiikkamoottori on hyvin tunnettu NVidian kehittämä fysiikkamoottori. PhysX:llä voi luoda 3D-ympäristöjä ja niiden sisällä toimivia objekteja, joilla on omat ominaisuutensa, kuten nopeus, suunta ja sijainti (NVIDIA n.d.). PhysX pystyy kehittyneeseen törmäyksen havaitsemiseen, jolloin kappaleitten voimat törmäyksen jälkeen ovat realistiset. Tämä mahdollistaa todellisten liikeratojen ennakoimisen, jolloin simuloimalla useita kertoja voidaan esimerkiksi selvittää, kuinka usein laite tekee virheellisen liikkeen ja millaisia vaikutuksia sillä on ympäristöön. Toinen tärkeä ominaisuus PhysX:ssä on nivelliikeradat. PhysX:sisältää useita erilaisia valmiita niveltyyppejä kuten sarananivel ja pallonivel. PhysX tukee geometriakyselyitä, joilla voidaan selvittää, onko objekteja päällekkäin tai missä vaiheessa objekti osuu toiseen, jos se jatkaa samaa rataa. Box2D on 2D-simulaatiokirjasto, jolla kahdessa ulottuvuudessa kulkeville kappaleille voidaan simuloida realistiset liikeradat ja vuorovaikutukset (Box2D n.d). Se toimii samoin 2D-ympäristölle, kuin PhysX 3D-ympäristölle. Unity Physics on paketti, jolla pelimoottoriin voidaan tuoda dataorientoitu suunnittelumalli. Datan kontrollointi entiteeteillä helpottaa niiden kontrollointia. Havok-fysiikkamoottori on Unity Physicsin DOTS-teknologian (Data-Oriented Technology Stack) päälle rakennettu erittäin vankka ja nopeuteen optimoitu fysiikkamoottori (Zarco et al 2021), joka tuottaa varmemmin realistia hankalia pomppuja ja kitkaa kuin PhysX. PhysX:llä on kuitenkin yksi parhaista suorituskyvyistä vakailla rajoituksilla.

Chaos Physics on betavaiheessa oleva Unreal Enginen kehittämä kevyt fysiikkamoottori (Unreal Engine, 2022). Chaos Physics korvaa aikaisemmin käytössä olleen PhysX:n Unreal Enginen fysiikkamoottorina. Chaos Physicsillä on hyvin samanlaiset ominaisuudet kuin PhysX:llä ja sen lisäksi sillä on kehittynyt tuhoutumissysteemi, jolla voi simuloida erittäin realistisesti esimerkiksi rakennuksen purkamista.

Digitaalisten kaksosten simulointi- ja optimointialustana käytetään lähtökohtaisesti PC:tä, sillä kyseessä on suhteellisen raskas ohjelma, jota ei välttämättä puhelimella tai verkon yli ole tehokasta tai tarkoituksenmukaista ajaa. Erityisesti simuloinnissa voidaan kuitenkin pidemmällä tähtäimellä suunnitella, halutaanko sovelluksen visualisointi järjestää mukana kannettavaksi, kuten mobiililaitteelle. Pelimoottorit tekevät tuotteita hyvin pelikonsoleille, kuten XBOX ja PlayStation, joissa 3D-ympäristö voidaan ottaa käyttöön virtuaalisessa todellisuudessa. Useimmilla pelimoottoreilla alustavaihtoehdot ovat hyvin monipuoliset, sillä perinteisiä pelejä niillä tuotetaan lähes jokaiselle alustalle.

Lisenssin hinta vaikuttaa pelimoottorin käytön kannattavuuteen. Unity3D perustuu lisenssimaksuihin. Ilmaisella versiolla saa pelimoottorin, mutta maksua vastaan sen saa täydennetyillä ominaisuuksilla, kuten Havoc-fysiikkamoottorilla ja lähdekoodilla. Unreal

Engine perustuu rojaltimaksuihin. Pelimoottoria saa käyttää ilmaiseksi, mutta ylittäessään yhden miljoonan USD:n bruttotuoton Epic Games ottaa 5 % rojaltin. Amazon O3DE on ilmainen avoimeen lähdekoodiin perustuva pelimoottori. O3DE käyttää kuitenkin osittain kolmannen osapuolen lisäosia, kuten tekoälykomponentteja sisältävän Kythera AI:n. O3DE:n käyttö on siis lähtökohtaisesti halvinta, ellei huomaa tarvitsevansa myöhemmin jotakin plug-in komponenttia, jonka käyttö maksaa.

Unity Technologies tarjoaa yrityksille Unity Industrial collection -pakettia, jossa tulee mukana kaikki DT:n luomiseen tarvittavat työkalut. Unity Pro -ohjelmiston lisäksi se sisältää Pixyz-lisäosan, jolla 3D-dataa on helppo tuoda ja optimoida sekä SystemGraphin, jolla voi mallintaa DT:n käyttäytymistä. Tällä paketilla pitäisi tehtaan kloonaaminen digitaaliseksi kaksoseksi onnistua tekemättä suuria integraatioita, sillä komponentit ovat räätälöity digitaalisen tehtaan suunnitteluun ja yhdessä toimimiseen.

AirSim on Unreal Enginen päälle rakennettu monen alustan simulaattori pääasiassa lennokeille, mutta sillä voi simuloida myös maassa olevia ajoneuvoja sekä muita objekteja (Microsoft n.d.). AirSimillä on simuloitu lennokkeja sekä liikenteenseurausympäristöjä. AirSim on Unreal Enginelle tehty, mutta siitä on kehitteillä versio myös Unitylle.

O3DE ei sisällä valmista viitekehystä digitaalisille kaksosille eikä sille löydy virallista lisäosaa. Tämä ei estä digitaalisten kaksosten tekemistä O3DE:llä, mutta työnkulku on varmasti hitaampaa, monimutkaisempaa ja tiedonhaku vaikeampaa.

Näiden ominaisuuksien perusteella ainakin Unity3D ja Unreal Engine tarjoavat perustan toteuttaa digitaalisia kaksosia tehokkaasti. O3DE:n käyttöönoton monimutkaisuus tekee siitä arveluttavan vaihtoehdon teolliseen käyttöön, sillä hankinta vaatisi huomattavasti suuremman määrän tutkimusta kuin muilla tutkituilla pelimoottoreilla. Unity3D ja Unreal Engine myös tarjoavat näyttöä toteutuksesta, jota O3DE:ltä ei löydy. O3DE siis vaikuttaisi olevan riskialtis epäonnistumiselle, vaikka muuten ominaisuudet ei vaikuttaisi jäävän jälkeen muille pelimoottoreille.

## 4.2 Tekniset ominaisuudet

Pelimoottorien teknisiä ominaisuuksia tarkastellessa otetaan huomioon kytkeytymiseen ja ulkonäköön liittyvät ominaisuudet. Nykyään pelimoottorit riittävät totisten pelien toteuttamiseen, sillä ne ovat edistyneet jo tuottamaan lähes hyperrealistisia pelejä. Useimpien digitaalisten kaksosten ei todennäköisesti kuitenkaan tarvitse olla niin tarkkoja, sillä toiminnallisuus on optimoinnissa ja simuloinnissa tärkeämpää, kuin erittäin tarkka kuvanlaatu. Verkkoon kytkeytyminen voi kuitenkin olla toivottu ominaisuus esimerkiksi

reaaliaikaisten simulointien tapauksessa. Tässä luvussa olevat tekniset ominaisuudet ei kuitenkaan ole kriittisinä määrittävinä tekijöinä digitaalisten kaksosten luomiselle.

| | **Unity3D** | **Unreal Engine** | **O3DE** |
|---|---|---|---|
| **Verkkoon kytkeytyminen** | Kyllä | Kyllä | Kyllä |
| **Teksturointi** | Basic, Bumbmapping, Procedural | Basic, Multitexturing, Bumbmapping, Procedural | N/A |
| **Piirtotarkkuus** | Korkea | Erittäin korkea | Korkea |

Taulukko 2. *Pelimoottorien tekniset ominaisuudet (Zarco et al. 2021)*

Taulukossa 2 on lueteltu pelimoottoreiden teknisiä ominaisuuksia. Verkkoon kytkeytymien pelien kohdalla tarkoittaa sitä, että voidaan luoda usea pelaajan pelejä. Totisten pelien kohdalla ei välttämättä ole olennaista, että useat toimeksiantajat voivat olla vuorovaikutuksessa keskenään simuloidussa ympäristössä. Reaaliaikaisen simuloinnin tapauksessa kuitenkin laitteista kerätty data on saatava kulkemaan verkon yli, jolloin verkkoon kytkeytymistä tarvitaan. Useimmissa pelimoottoreissa tämä on mahdollista, mutta joissakin vain rajatusti (Zarco et al. 2021). Kuten taulukosta 2 nähdään, valituissa pelimoottoreissa rajoitteita ei ole.

Teksturointi vaikuttaa DT:n ulkonäköön. Jokainen valituista pelimoottoreista toteuttaa tunnistettavan kaksosen, mutta lopputuloksen ulkonäkö vaihtelee. Erilaisilla teksturointimenetelmillä on mahdollista saada aikaan realistisempia pintoja. Tunnetuimmat pelimoottorit on kehitetty luomaan korkealaatuisia pelejä verrattuna vähemmän tunnettuihin pelimoottoreihin. Toiminnallisten sovelluksen tapauksessa jokaisen pelimoottorin pitäisi saada tyydyttävän näköisiä tuotteita aikaiseksi. Unreal Engine on kuitenkin pyrkinyt olemaan tarkkuudeltaan kehittynein pelimoottori markkinoilla ja pystyykin lähes hyperrealistisiin tarkkuuksiin.

Korkea piirtotarkkuus on tyypillisesti tärkeää silloin, kun simuloinnista opittu asia halutaan siirtää virtuaalisesta ympäristöstä fyysiseen asetelmaan (Petridis et al. 2010). Mitä tarkempi kuvaus todellisuudesta virtuaalinen ympäristö on, sitä helpompi sitä on soveltaa ympäristön ulkopuolella. Jos yleiskuvan simulointi on tärkeämpää kuin yksityiskohtien, huonommallakin piirtotarkkuudella päästään hyvään lopputulokseen.

## 4.3 Perusominaisuudet

Perusominaisuuksilla tarkoitetaan pelimoottorin käyttöön liittyviä ominaisuuksia, kuten käyttämisen helppous ja pelimoottorin käyttöhinta. Perusominaisuuksia tarkastelemalla voidaan selvittää, voiko pelimoottoria käyttää, kun taas teknisiä ominaisuuksia tarkastelemalla selviää, minkälaisia tuotteita sillä voidaan luoda.

Perusominaisuudet ovat totisissa peleissä todennäköisesti enemmän määrittäviä tekijöitä, kuin tekniset ominaisuudet, sillä niiden on oltava yhteneviä osaajien ja tarpeiden kanssa. On esimerkiksi järkevää käyttää pelimoottoria, jonka oppimiskäyrä ei ole kovin jyrkkä, jos kehittäjät eivät ole jo valmiiksi jonkun vaikeamman pelimoottorin ammattilaisia. Toisekseen 2D/3D soveltuvuudella on merkitystä, jos sovelluskohteet voivat olla myös 2D:nä. Suurimmalla osalla pelimoottoreista löytyy 3D-mallinnuksen mahdollisuus, mutta esimerkiksi ylhäältä kuvattuja 2D-optimointimalleja voi olla tehokasta tehdä sarjatuotantona pienempiin tarpeisiin.

| | **Unity3D** | **Unreal Engine** | **O3DE** |
|---|---|---|---|
| **2D/3D** | 2D/3D | *3D* | 2D/3D |
| **Tekoäly** | Botit ja FSM | FSM | *N/A* |
| **Tekninen ja yhteisön tuki** | *Hyvä/Hyvä* | *Hyvä/Hyvä* | *Hyvä/Huono* |
| **Helppokäyttöisyys** | *Helppo käyttää* | *Melko helppo käyttää* | *Keskivaikea* |
| **CAD** | .fbx .dae .3ds .dxf .obj | .fbx .obj | .fbx |

Taulukko 3. *Pelimoottorien perusominaisuudet (Zarco et al. 2021)*

Taulukossa 3 on lueteltu pelimoottoreiden perusominaisuuksia. Useimmat pelimoottorit pystyvät hyödyntämään kolmea ulottuvuutta ja vain harvat pelimoottorit kykenevät vain 2D-pelien tekemiseen. Vain Unity3D:llä voidaan tehdä 2D-tuotteita.

Ohjaamisen automatisointi on mahdollista pelimoottorin tukemalla tekoälyllä. Unity ja Unreal tukevat rajallisia tilakoneita ja Unity sen lisäksi vielä botteja. Tilakoneilla laitteet saadaan käyttäytymään halutun prosessin mukaisesti helposti, sillä simuloinnissa ei tarvitse todennäköisesti ottaa huomioon kovin epävarmoja tilanteita. Amazonin O3DE:n tukea tekoälylle ei olla ilmoitettu.

Uutta ohjelmistoa käyttäessä tekninen tuki saattaa olla projektin estymiselle määrittävä tekijä. Usein tulee tilanne vastaan, jolloin dokumentaatio ei ole riittävä, vaan on käännyttävä osaavan ihmisen puoleen. Tällöin apuna on tekninen tuki sekä yhteisön tuki. Teknisessä tuessa ongelmaan vastaa mahdollisesti pelimoottorin luomisessa mukana

ollut henkilö ja yhteisön tuessa vastauksen löytää usein nettikeskusteluista, jossa joku muu on etsinyt vastauksen samaan ongelmaan. Käytön helppouteen vaikuttaa myös se, kuinka helpoksi ohjelmointi pelimoottorin sisällä on tehty. Pelimoottorit on suunniteltu helpottamaan pelien tekemistä ja pienempään kynnykseen luomaan pelejä ilman aikaisempaa kokemusta, mutta osa niistä silti vaatii korkeampaa osaamista ohjelmoinnissa. Unityn ja Unreal Enginen käyttövaikeutta on luokiteltu helpoksi Zarcon ja muiden mukaan (2021), kun taas O3DE on luokiteltu keskivaikeaksi. Tarkemmin katsottuna, Unity3D:n pitäisi olla Unreal Engineä helpompi käyttää, sillä Unity3D:n skriptauskieliä ovat C# ja JavaScript, jotka ovat korkeamman tason kieliä, kuin Unreal Enginen käyttämä C++. Korkeamman tason kielet ovat abstraktimpia, jolloin kehittäjän tarvitsee ottaa vähemmän tietokoneen arkkitehtuuriin liittyviä asioita huomioon. Unreal Enginen Blueprints on korkeamman tason skriptauskieli, mutta se pääsee huonommin käsiksi moottorin resursseihin kuin C++.

Mallien integrointi 3D-mallinnusohjelmasta pelimoottoriin on lähtökohtaisesti sama kaikilla pelimoottoreilla. Yleisesti hyväksytty tuontimuoto on. fbx, joka sisältää 3D-mallin lisäksi myös mallin metatiedot. Useimmat pelimoottorit kuitenkin pystyvät vastaanottamaan useita muitakin tyyppejä sisäänrakennetuilla konvertointityökaluilla (Unity, 2021: Epic Games n.d.).

Perusominaisuudet tukevat DT:n rakentamista. Kattavat vaihtoehdot taulukossa 3. antavat monipuolisemmat vaihtoehdot DT:lle sekä tekee luomisprosessista huomattavasti helpompaa ja nopeampaa. Erityisesti kattava tuki estää pysähdykset ja virheiden etsintä tuottaa vähemmän hankaluuksia.

# 5. YHTEENVETO

Tässä tutkimuksessa selvitin, mitä ominaisuuksia digitaalinen kaksonen vaatii pelimoottorilta ja miten kuluttajamarkkinoiden suurimmat pelimoottorit toteuttavat ne vaatimukset. Pelimoottorit ovat jo valmiiksi melko monimutkaisia työkaluja siitä huolimatta, että ne on luotu helpottamaan pelien tekemistä. Kun siihen lisää vielä tarpeen toteuttaa hyödyllisiä ympäristöjä sujuvalla työnkululla ja loivalla oppimiskäyrällä, ei mikä tahansa työkalu välttämättä sovi. Helpointa olisi, jos markkinoilla olisi vakiintuneita palveluntarjoajia, jotka tarjoisivat monipuolisia digitaalisen kaksosen luomiseen

tarkoitettuja työkaluja ja valmiita viitekehyksiä, mutta niitä ei vielä ole vakiintunut. Suurin osa pelimoottoreilla rakennetuista digitaalisista kaksosista on tehty tutkimukseen liittyen, eikä loppukäyttöön tarkoitettuja tuotteita ole vielä valtavasti. Tämän vuoksi selvää tietoa ei vielä ole, mikä olisi selvästi järkevin tapa lähteä hyödyntämään digitaalisia kaksosia ja pelimoottoreita yhdessä.

Otin tutkimukseen mukaan kolme perustarpeet täyttävää pelimoottoria. Valinnasta jätettiin pois pelimoottorit, jotka eivät esimerkiksi kyenneet 3D-simulointiin. Valitut pelimoottorit kykenivät ensisilmäyksen perusteella toteuttamaan digitaalisia kaksosia, mutta vielä ei ollut varmaa, ovatko ne pitkällä tähtäimellä hyviä alustoja teolliseen käyttöön. Vertailusta selvisi, että tarjolla olevat ennakkosuosikit Unity3D ja Unreal Engine tukevat parhaiten digitaalisten kaksosten tuottamista. Unity3D mainostaa tarjoavansa kokonaisuuden kattavaa paketteja digitaalisten kaksosten rakentamiseen, jolla koko työnkulku on optimoitu tehdasasetuksen mallintamiseen usean eri työkalun integraatiolla. Unreal Enginen päälle on puolestaan rakennettu AirSim-simulointialusta, jota on jo viisi vuotta testattu lennokkien simuloinnissa. Unity3D tulee myöhemmin tukemaan AirSimiä myös, joten Unityn käyttäminen ei sulje sitä vaihtoehtoa kokonaan pois. Amazonin O3DE:tä ei olla käytetty missään merkittävässä tehdasympäristön simulointitutkimuksessa, eikä O3DE:n dokumentaatiokaan mainosta viitekehystä digitaalisille kaksosille. O3DE voisi testatessa yllättää kyvyillään, mutta kirjallisuus ei viittaa siihen, että se olisi millään osa-alueella parempi vaihtoehto kuin Unity3D tai Unreal Engine.

# LÄHDELUETTELO


Adamenko, D.; Kunnen, S.; Pluhnau, R.; Loibl, A.; Nagarajah, A. (2020). Review and comparison of the methods of designing the Digital Twin. (ss. 27-32). Procedia CIRP. Noudettu osoitteesta https://doi.org/10.1016/j.procir.2020.02.146

Box2D. (n.d.). *Overview*. Haettu: 24.1.2022 osoitteesta https://box2d.org/documentation/index.html

CGIFURNITURE. (n.d.). *3D File Formats*. Haettu: 25.11.2022 osoitteesta https://cgifurniture.com/3d-rendering-guide/common-3d-file-formats-product-cgi/

Dehghanimohammadabadi, M.; Belsare, S.; Thiesing, R. (2021). Simulation-optimization of digital twin. (ss. 1-10). Proceedings of the Winter Simulation Conference. Noudettu osoitteesta https://dl.acm.org/doi/10.5555/3522802.3523000

Epic Games. (n.d.a). *Chaos Physics Overview*. Haettu: 24.11.2022 osoitteesta https://docs.unrealengine.com/4.26/en-US/InteractiveExperiences/Physics/ChaosPhysics/Overview/

Epic Games. (n.d.b). *Datasmith Supported Software and File Types*. Haettu: 24.11.2022 osoitteesta https://docs.unrealengine.com/4.26/en-US/WorkingWithContent/Importing/Datasmith/SupportedSoftwareAndFileTypes/

Gregory, J. (2018). *Game engine architecture* (Kolmas painos p.). A K Peters/CRC Press.

Hyre, A.; Harris, G.; Osho, J.; Pantelidakis, M.; Mykoniatis, K.; Liu, J. (2022). *Digital twins: Representation, Replication, Reality, and Relational (4Rs)* (Vol. 31 p.). Manufacturing Letters. Noudettu osoitteesta https://doi.org/10.1016/j.mfglet.2021.12.004

Jones, D.; Snider, C.; Nassehi, A.; Yon, J.; Hicks, B. (2020). *Characterising the Digital Twin: A systematic literature review* (Vol. 29 p.). CIRP Journal of Manufacturing Science and Technology. Noudettu osoitteesta https://doi.org/10.1016/j.cirpj.2020.02.002

Kritzler, M.; Funk, M.; Michahelles, F.; Rohde, W. (2017). *The virtual twin: controlling smart factories using a spatially-correct augmented reality representation.* Proceedings of the Seventh International Conference on the Internet of Things. Noudettu osoitteesta https://dl.acm.org/doi/10.1145/3131542.3140274

Microsoft. (n.d.). *AirSim announcement: This repository will be archived in the coming year*. Haettu: 24.11.2022 osoitteesta https://microsoft.github.io/AirSim/

NVIDIA. (n.d.). *A brief overview of PhysX*. Haettu: 24.11.2022 osoitteesta https://gameworksdocs.nvidia.com/PhysX/4.0/documentation/PhysXGuide/Manual/Introduction.html#a-brief-overview-of-physx

Petridis, P.; Dunwell, I.; Freitas, S.; Panzoli, D. (2010). *An Engine Selection Methodology for High Fidelity Serious Games.* IEEE 2010 Second International Conference on Games and Virtual Worlds for Serious Applications. doi:https://ieeexplore.ieee.org/abstract/document/5460160

Qi, Q.; Tao, F.; Zuo, Y.; Zhao, D. (2018). Digital Twin Service towards Smart Manufacturing. (ss. 237-242). Procedia CIRP. Haettu: 1.10.2022 osoitteesta https://www.sciencedirect.com/science/article/pii/S2212827118302580

Tao, F.; Xiao, B.; QI, Q.; Cheng, J.; Ji, P. (2022). *Digital twin modeling* (Vol. 64 p.). Journal of Manufacturing Systems. Noudettu osoitteesta https://doi.org/10.1016/j.jmsy.2022.06.015

Uhlemann, T.; Schock, C.; Lehmann, C.; Freiberger, S.; Steinhilper, R. (2017). *The Digital Twin: Demonstrating the Potential of Real Time Data Acquisition in Production Systems* (Vol. 9 p.). Procedia Manufacturing. Noudettu osoitteesta https://doi.org/10.1016/j.promfg.2017.04.043

Unity. (2021). *Supported Model file formats*. Haettu: 24.112022 osoitteesta https://docs.unity3d.com/2020.1/Documentation/Manual/3D-formats.html

Zarco, L.; Siegert, J.; Schlegel, T.; Bauernhansl, T. (2021). *Scope and delimitation of game engine simulations for ultra-flexible production environments* (Vol. 104 p.). Procedia CIRP. Noudettu osoitteesta https://doi.org/10.1016/j.procir.2021.11.133