SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Jenna Jääskeläinen

Simulaatio- ja käyttöliittymäkehitysprosessien vertailu ja uudistaminen

Opinnäytetyö
Kevät 2025
Insinööri (AMK), Automaatiotekniikka

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Tutkinto-ohjelma: Insinööri (AMK), Automaatiotekniikka

Suuntautumisvaihtoehto: Koneautomaatio

Tekijä: Jenna Jääskeläinen

Työn nimi: Simulaatio- ja käyttöliittymäkehitysprosessien vertailu ja uudistaminen

Ohjaaja: Marko Hietamäki

Vuosi: 2025 Sivumäärä: 42

Tämän työn toimeksiantajana toimi Pesmel Oy, joka valmistaa automaattisia varastointi-, logistiikka- ja pakkausjärjestelmiä paperi- ja sellu, metalli- sekä rengasteollisuuden yrityksille. Yrityksellä oli tarve kehittää sen nykyistä simulointi- ja käyttöliittymäkehitysprosessia projektikohtaisen teollisuusympäristön visualisoinnin osalta suoraviivaisemmaksi ja yksinkertaisemmaksi muuttamalla toteutusta selkeämmäksi. Työn tavoite oli vertailla nykyistä prosessia uuteen, Unity-pelimoottorilla kehitettyyn versioon ja selvittää, tuoko toimintatapojen muutos toivottuja hyötyjä prosessin toteutukseen.

Aluksi selvitettiin yrityksen nykyinen toimintamalli ja suunniteltiin uuden prosessin rakenne. Sen pohjalta luotiin Unity-ympäristössä 3D-malliin perustuva WMS-simulaatio ja myyntivisualisaatio. WMS-simulaatio vastaanotti dataa WMS-backendin simulaatiomoduulilta WebSocket-protokollan ja REST-rajapinnan avulla, kun taas myyntivisualisaatio toteutettiin animoimalla laitteiden liikkeet erillisillä ohjelmakoodeilla.

Lopputuloksena kehitettiin WMS-simulaatio ja myyntivisualisaatio sekä vertailtiin nykyistä ja uutta toimintamallia. WMS-simulaatio saatiin upotettua WMS-selainversioon Unityn WebGL-ominaisuutta hyödyntäen. Lopputulos vahvisti konseptin toimivuuden, jonka perusteella sitä voidaan jatkokehittää. Tulokset osoittivat, että uuden toimintatavan käyttöönotto edellyttäisi muutoksia yrityksen sisäisiin toimintatapoihin.

[1] Asiasanat: simulointi, käyttöliittymät, digitaalinen kaksonen, varastointi

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Degree programme: Bachelor of Engineering, Automation Engineering

Specialisation: Machine Automation

Author: Jenna Jääskeläinen

Title of thesis: Comparison and improvement of simulation and user interface development process

Supervisor: Marko Hietamäki

Year: 2025 Number of pages: 42

The thesis was commissioned by Pesmel Oy, a company specializing in automatic warehousing, logistics, and packing solutions for pulp and paper, as well as tire and metal industries. The aim of the thesis was to compare the current simulation and user interface development process in terms of project-specific visualization of the industrial environment of the company with a new process implemented using a Unity game engine. The goal was to simplify and clarify the process, making it more straightforward. With the help of the comparison, it could be evaluated whether the new approach would bring the desired results.

Firstly, the current course of action was investigated, and the structure for the new process was discussed. The practical implementation was started by creating a WMS simulation in the Unity environment based on a 3D model of the warehouse. The simulation performed actions based on the data retrieved from the WMS-Backend to Unity using WebSocket protocol and REST API. In addition to that, a simplified sales visualization was created using separate programming scripts.

As the result of the thesis, a WMS simulation and sales visualization were created using the Unity game engine, and the current and new courses of actions were compared. Unity's WebGL feature made it possible to publish the WMS simulation on the browser version of the WMS. The result was a proof of concept that can be further developed. However, the introduction of the new course of action would require internal changes in the practices of the company.

[1] Keywords: simulation, user interfaces, digital twin, warehousing

SISÄLTÖ

Opinnäytetyön tiivistelmä 1
Thesis abstract 2
SISÄLTÖ 3
Kuva-, kuvio- ja taulukkoluettelo 5
Käytetyt termit ja lyhenteet 7
1 JOHDANTO 8
1.1 Työn tausta 8
1.2 Työn tavoite 9
1.3 Työn rakenne 9
1.4 Yritysesittely 9
2 SIMULOINTI 11
2.1 Simuloinnista yleisesti 11
2.2 Simuloinnin hyödyt ja haitat 11
2.3 Simuloinnin vaiheet 12
2.4 Digitaalinen kaksonen 15
2.5 Simulointiohjelmistot 15
2.5.1 Visual Components 16
2.5.2 Unity 16
3 VARASTONHALLINTAJÄRJESTELMÄ 17
3.1 Johdanto varastonhallintajärjestelmiin 17
3.2 Pesmel WMS 18
3.3 Käyttökokemus ja käyttöliittymä 19
3.4 Graafinen käyttöliittymä 19
4 NYKYINEN SIMULOINTI- JA KÄYTTÖLIITTYMÄKEHITYSPROSESSI 20
4.1 Myyntisimulointi 20
4.2 WMS-käyttöliittymä 21
4.2.1 Layout-sivu 21
4.2.2 Warehouse-sivu 22

4.3 Kaksiulotteisen WMS-layoutin kehittäminen 23
4.4 WMS-simulointi 24
5 UUSI SIMULOINTI- JA KÄYTTÖLIITTYMÄKEHITYSPROSESSI 25
5.1 3D-mallinnus 25
5.2 Varastomallin tuominen Unity-ympäristöön 27
5.3 Mallin pilkkominen osiin 29
5.4 Myyntivisualisaatio 30
5.5 WMS-integraatio 30
6 TOIMINTAMALLIEN VERTAILU 33
7 TULOKSET JA POHDINTA 36
8 YHTEENVETO 38
LÄHTEET 39

Kuva-, kuvio- ja taulukkoluettelo

Kuva 1. Teollisuusympäristön visualisointi Visual Components -toteutuksena. 8

Kuva 2. WMS-käyttöliittymän layout-sivu, jossa varaston 2D-kuva ylhäältä kuvattuna. 22

Kuva 3. WMS-käyttöliittymän warehouse-sivu, jossa varaston 2D-kuva sivultapäin kuvattuna. 22

Kuva 4. Yksinkertaistettu 3D-malli varastohyllyn päätyrakenteesta. 25

Kuva 5. Hyllystöhissin alkuperäinen (vas.) ja muokattu 3D-malli. 26

Kuva 6. Esimerkki pääkokoonpanosta. 27

Kuva 7. Varastomalli Unity-ympäristössä. 28

Kuva 8. Prefabin purkaminen Unityssä. 29

Kuva 9. 3D-layout upotettuna WMS-selainversioon. 36

Kuva 10. 3D-layout sivusta päin kuvattuna (ortografinen kamera). 36

Kuvio 1. Simuloinnin vaiheet. 12

Kuvio 2. Tiedon kulkeminen ERP-, MES- ja WMS-järjestelmien välillä. 18

Kuvio 3. SVG-layoutin kehittämisen vaiheet. 24

Kuvio 4. Tiedon kulkeminen. 31

Kuvio 5. Uuden simulaatio- ja käyttöliittymäkehitysprosessin vaiheet. 32

Kuvio 6. Nykyinen toimintamalli. 33

Kuvio 7. Uusi toimintamalli. 34

Taulukko 1. Toimintamallien vertailu...37

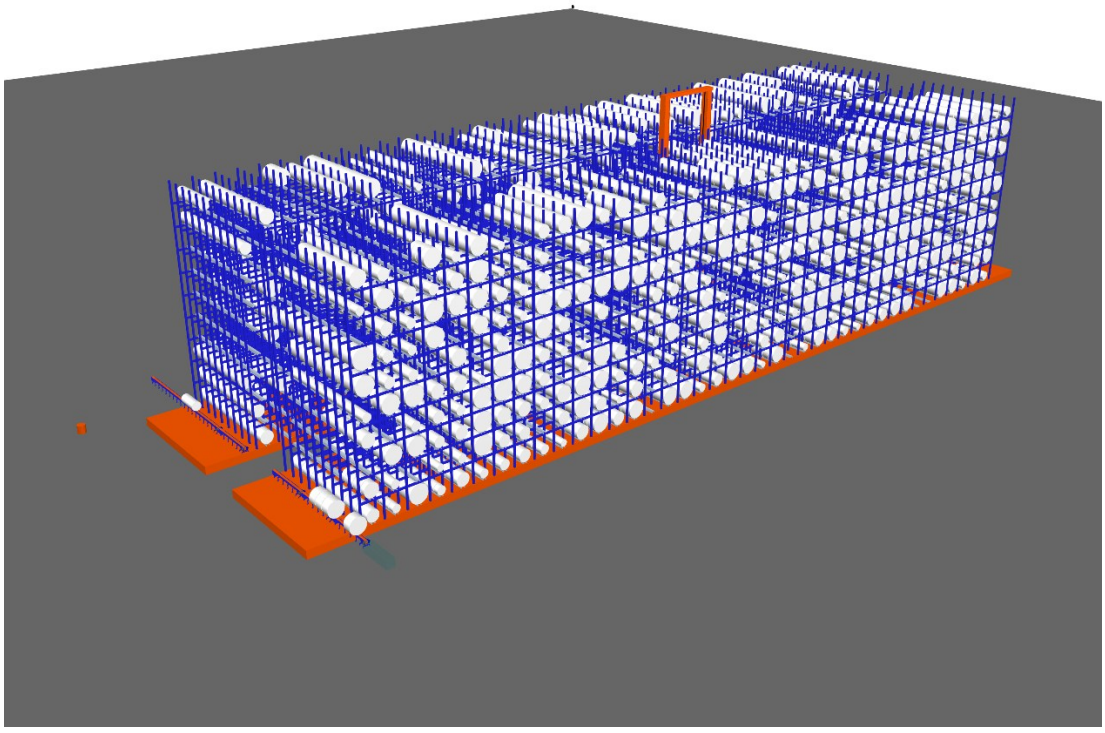
Käytetyt termit ja lyhenteet

API	Application Programming Interface, ohjelmointirajapinta. Mahdollistaa ohjelmistojen välisen kommunikaation.
Asset	Mikä tahansa resurssi tai sisältö Unity-ympäristössä.
Backend	Verkkosivun palvelinpuoli.
Frontend	Verkkosivun selainpuoli, jonka kanssa käyttäjä on vuorovaikutuksessa.
GameObject	Perusluokka kaikille Unityn sceneen kuuluville objekteille.
Prefab	Tallennettu GameObject-malli, joka sisältää itse GameObjectin, sen komponentit ja asetukset.
REST	Representational State Transfer. Arkkitehtuurityyli, jossa asiakas lähettää HTTP-pyynnön palvelimelle ja vastaanottaa tiedon yleensä JSON-muodossa.
Scene	Unity-projekti koostuu sceneistä, jotka sisältävät kaikki projektin elementit ja määrittävät sen toiminnan sekä visuaalisen sisällön.
SVG	Scalable Vector Graphics. XML-kieleen perustuva kaksiulotteisen vektorigrafiikan esittämistapa.
Unit	Unity-ympäristössä käytetty mittayksikkö. Oletuksena yksi unit vastaa yhtä metriä.
WebGL	Web Graphics Library. Selainpohjainen teknologia, joka mahdollistaa 3D- ja 2D-grafiikan renderöinnin verkkoselaimessa ilman lisäosia.
WebSocket	Protokolla, joka mahdollistaa kaksisuuntaisen, reaaliaikaisen viestinnän asiakkaan ja palvelimen välillä.
WMS	Warehouse Management System, varastonhallintajärjestelmä.

1 JOHDANTO

1.1 Työn tausta

Opinnäytetyön toimeksiantajalla, Pesmel Oy:llä, on tarve selvittää vaihtoehtoinen toimintatapa yrityksen simulointi- ja käyttöliittymäkehitysprosessille projektikohtaisen teollisuusympäristön visualisoinnin osalta, josta on esimerkki kuvassa 1. Simulointia hyödynnetään yrityksessä myyntivaiheessa sekä varastonhallintajärjestelmissä. Myynnin tueksi tuotetaan kahdentyyppisiä simulaatioita. Myyntivisualisaatiot ovat yksinkertaisempia simulaatioita, joiden tarkoitus on visualisoida järjestelmän toimintaa asiakkaalle. Varsinaisilla myyntisimulaatioilla kerätään järjestelmistä erinäistä statistiikkaa sekä suoritetaan niiden optimointia, validointia ja verifiointia. WMS-simulaatiot mallintavat todellisia järjestelmiä ja niiden reaaliaikaista toimintaa. WMS-simulointia tehdään myynti- tai WMS-kehitysvaiheessa, joissain tapauksissa molemmissa. Sen avulla on mahdollista validoida ja optimoida järjestelmiä. Lisäksi se auttaa ennakoimaan tilanteita ja helpottaa järjestelmien käyttöönottoa.



Kuva 1. Teollisuusympäristön visualisointi Visual Components -toteutuksena (Laurila, i.a.).

Yrityksen nykyisessä simulointi- ja käyttöliittymäkehitysprosessissa myyntisimulaatio on jäänyt myyntivaiheen jälkeen tarpeettomaksi ja WMS-simulaatioiden tekeminen teollisuusympäristön visualisoinnin osalta on ollut monivaiheinen sekä työläs prosessi. Lisäksi kilpailu varastoautomaation alalla on lisännyt painetta WMS 3D -käyttöliittymän kehittämiselle. Pesmelin tavoite on yksinkertaistaa prosesseja siirtymällä yhteen työkaluun, joka parhaassa tapauksessa toisi synergiaetuja.

1.2 Työn tavoite

Tämän työn tavoitteena on vertailla Pesmel Oy:n aiemmin toteuttamaa simulointi- ja WMS-käyttöliittymäkehitysprosessia Unity-pelimoottorilla suoritettavaan prosessiin. Vertailun avulla saadaan käsitys siitä, tuoko uusi toimintamalli etuja prosessin toteutukseen aiempaan verrattuna.

1.3 Työn rakenne

Työn johdannossa esitellään työn toimeksiantaja, tausta ja tavoite. Teoriaosuus koostuu luvuista kaksi ja kolme. Luvussa kaksi kerrotaan simuloinnista, sen hyödyistä ja haitoista sekä vaiheista, digitaalisesta kaksosesta ja simulointiohjelmistoista. Luvussa kolme käsitellään varastonhallintajärjestelmiä ja käyttöliittymiä. Työn käytännön osuus on kuvattuna luvuissa neljä ja viisi, joista luvussa neljä kuvataan nykyisen ja luvussa viisi uuden toimintamallin toteutuksen vaiheet. Toimintatapojen vertailu on luvussa kuusi. Lukuun seitsemän on koottuna työn tulokset ja pohdintaa. Luku kahdeksan sisältää yhteenvedon työstä.

1.4 Yritysesittely

Pesmel Oy on Kauhajoella vuonna 1978 perustettu automatisoitujen varastointi-, logistiikka- ja pakkausjärjestelmien valmistaja (Pesmel, 2024). Pesmelin pääkonttori sijaitsee Kauhajoella, jonka lisäksi sillä on toimipaikat Seinäjoella ja Helsingissä sekä tytäryhtiöitä ja laaja edustajistoverkosto maailmanlaajuisesti. Vuonna 2023 Pesmelin liikevaihto oli 56,1 miljoonaa euroa ja se työllisti n. 160 henkilöä (Suomen asiakastieto, i.a.).

Pesmelin asiakaskunta koostuu paperi- ja sellu- sekä metalli- ja rengasteollisuuden yrityksistä, joille se on yli nelikymmenvuotisen taipaleensa aikana toimittanut yli 400 järjestelmää viidelle mantereelle (Pesmel, 2024). Pesmelin *Material Flow How*® -konsepti tarjoaa kokonaisvaltaisen ja räätälöidyn ratkaisun vastaamaan jokaisen asiakkaan tapauskohtaisia tarpeita ja vaatimuksia.

2 SIMULOINTI

2.1 Simuloinnista yleisesti

Simulaatio on jäljitelmä todellisesta järjestelmästä ja sen muuttuvista prosesseista (Bangsow, 2020, s. 2). Järjestelmän voidaan määritellä olevan erillinen joukko toisiinsa liittyviä osia, laitteita tai kokonaisuuksia, kun taas malliksi kutsutaan yksinkertaistettua jäljennöstä suunnitellusta tai todellisesta järjestelmästä, joka sisältää sen prosessit toisessa järjestelmässä. Malli eroaa alkuperäisestä tärkeiden ominaisuuksien osalta vain ennalta määriteltyjen toleranssien rajoissa.

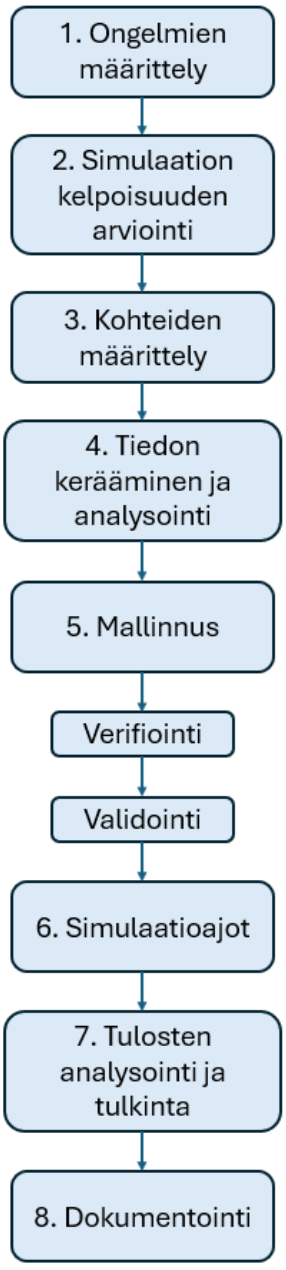
Simulaatiomallin tarkoitus on siis kerätä tietoa tietyn järjestelmän toiminnasta ajan suhteen (Rossetti, 2016, s. 11). Simuloinnissa mallille luodaan keinotekoinen historia, jota tarkkaillaan, jotta voidaan tehdä johtopäätöksiä sen todellisen vastineen toimintaominaisuuksista (Banks, 2005, s. 3). Simulaatiomalli on usein joukko olettamuksia liittyen järjestelmän toimintaan. Ne esitetään matemaattisina, loogisina ja symbolisina suhteina entiteettien eli tarkasteltavien kohteiden välillä. Mallin avulla pystytään vastaamaan monipuolisesti todellista järjestelmää koskeviin ”mitä jos” -kysymyksiin. Kun halutaan tietää mahdollisten muutosten vaikutus jo olemassa olevaan järjestelmään, ne voidaan simuloida. Tällöin niiden vaikutus sen toimintaan on ennakoitavissa. Simulointia on mahdollista hyödyntää myös suunnitteluvaiheessa, kun halutaan saada selville uuden järjestelmän suorituskyky tietyissä olosuhteissa.

2.2 Simuloinnin hyödyt ja haitat

Simuloinnin suurin hyöty on sen tuomat taloudelliset säästöt (Bandyopadhyay & Bhattacharya, 2014, s. 14). Se helpottaa isojen järjestelmien analysoimista, ja simulaatiotutkimuksen tulokset vastaavat hyvin läheisesti todellisuutta, koska niissä käytetään todellista dataa. Simulaation avulla voidaan kokeilla erilaisia skenaarioita ja arvioida kriittisten tapahtumien vaikutusta tutkittavaan kohteeseen ilman riskejä. Simulointi mahdollistaa myös järjestelmän käytöksen tutkimisen pitkillä aikaväleillä ja simulaation nopeutta muuttamalla voidaan tarkastella sen toimintaa yksityiskohtaisemmin.

Vaikka simuloinnilla on monia hyötyjä ja se on riskitön tapa tutkia järjestelmän toimintaa, on sillä myös haittoja (Bandyopadhyay & Bhattacharya, 2014, s. 14). Simulaatiojärjestelmän lähtökulut voivat olla korkeat ja muuttujien keskinäisen vuorovaikutuksen tutkiminen kallista. Lisäksi simulaatiotutkimuksen tekeminen vaatii laajaa osaamista ja ymmärrystä.

2.3 Simuloinnin vaiheet



Kuvio 1. Simuloinnin vaiheet (Bangsow, 2020, s. 2; Rossetti, 2016, s. 54; Banks, 2005, s. 361).

Simulointi voidaan jakaa karkeasti kahdeksaan vaiheeseen kuvion 1 mukaisesti. Simulaatioprosessi alkaa ongelmien määrittelyllä, jossa simulaatiolle asetetaan vaatimukset ja määritetään tarkasteltavat ongelmat. Toinen vaihe on simulaation kelpoisuuden arviointi. Tässä vaiheessa päätetään, onko simulointi oikea lähestymistapa aiemmin määritettyjen ongelmien ratkaisuun (Bangsow, 2020, s. 2; Banks, 2005, s. 14).

Kolmas vaihe on kohteiden määrittely (Bangsow, 2020, s. 3). Tarkasteltavien kohteiden kokonaisuus koostuu usein pääkohteesta, joka jakautuu vuorovaikutuksessa toistensa kanssa oleviin osatavoitteisiin. Simuloinnissa yleisiä tarkasteltavia kohteita voivat olla mm. hyötysuhteiden maksimointi ja käsittelyaikojen tai varaston minimointi. Kaikista aluksi määritellyistä tarkastelukohteista tulee voida kerätä analysoitavaa statistiikkaa jokaiselta simulaatioajolta, joka vaatii simulaatiomallilta tiettyä tarkkuutta. Tutkittavat kohteet määrittelevät simulaation laajuuden.

Neljännessä vaiheessa kerätään ja analysoidaan simulaation kannalta olennaiset tiedot (Bangsow, 2020, s. 3). Tarvittavat tiedot kerätään tarkastelun kohteena olevasta todellisesta järjestelmästä (Banks, 2005, s. 307). Jos siitä ei ole mahdollista kerätä tietoa esimerkiksi resurssien puutteen vuoksi, voidaan tehdä valistuneita arvauksia perustuen asiantuntijoiden arvioihin. Kerätyn datan perusteella simulaatiolle määritellään tietyt parametrit. Simulaation kannalta merkittävä data voidaan ryhmitellä esimerkiksi tekniseen, organisaatiota koskevaan tai järjestelmän kuormitusta koskevaan dataan (Bangsow, 2020, s. 3).

Viides vaihe on simulaatiomallin kehitys ja testaus (Bangsow, 2020, s. 3). Aluksi simuloitavasta järjestelmästä luodaan yleiskuva, jonka avulla tehdään päätös simulaation tarkkuudesta. Riippuen siitä, kuinka tarkka sen tulee olla, voidaan joitain osa-alueita yksinkertaistaa.

Mallinnuksen ensimmäiseen vaiheeseen kuuluu analyysi ja yksinkertaistus. Systeemianalyysissä järjestelmä puretaan aiemmin määriteltyjen tutkimustavoitteiden mukaisesti osiin ja yksinkertaistetaan sisältämään vain simulaation toiminnan kannalta välttämättömät ominaisuudet. Tämän jälkeen simulaatiomalli kehitetään ja testataan. Malli verifioidaan, jotta voidaan varmistua simulaation tarkoituksenmukaisesta toiminnasta (Rossetti, 2016, s. 54). Se voi sisältää esimerkiksi virheiden etsimistä koodista tai simulaation toistettavuuden

varmistamisen. Mallin toiminnan validointi suoritetaan, jotta voidaan määrittää, vastaako se riittävän tarkasti todellista järjestelmää. Se kattaa prosessin, jossa simulaatiomallia ja sen toimintaa verrataan todellisuuteen (Banks, 2005, s. 361). Prosessi, jossa mallia verrataan toistuvasti todelliseen vastineeseen ja tehdään siihen tarvittavia muutoksia vertailun perusteella, on nimeltään kalibrointi. Vertailu todellisuuteen voi tapahtua subjektiivisesti, jolloin käytetään ihmisten tietotaitoa, tai objektiivisesti, jolloin verrataan todellisen järjestelmän ja mallin tuottamaa toisiaan vastaavaa dataa.

Mallinnusvaiheen jälkeen aloitetaan simulaatioajot (Bangsow, 2020, s. 4). Niissä suoritetaan kokeiluja, joilla saadaan tuloksia projektin päämääriin ja tavoitteisiin liittyen (Rossetti, 2016, s. 54). Kokeellisilla simulaatioajoilla tuotetaan vertailutilastoja nykyisen järjestelmän toiminnasta. Simulaatiomalli muutetaan vastaamaan mahdollista skenaariota, ja ajo suoritetaan uudelleen. Tätä prosessia toistamalla saadaan objektiivisesti vertailukelpoisia tuloksia vaihtoehtoisista skenaarioista. Joissain tapauksissa mallin suorituskykyyn vaikuttaa useita eri tekijöitä, jolloin voi olla tarpeen käyttää erilaisia tulosanalyysimenetelmiä, kuten eräajoja. Jokaiselta ajolta tulee dokumentoida sisään- ja ulostulodata sekä käytetyt parametrit (Bangsow, 2020, s. 5).

Simulaatioajojen jälkeen niistä saadut tulokset analysoidaan ja tulkitaan (Bangsow, 2020, s. 5). Jos tulokset ovat ristiriidassa ennen simulointia tehtyjen olettamuksien kanssa, tulee analysoida ne tekijät, jotka voivat vaikuttaa niihin. Monimutkaisissa järjestelmissä on usein tarpeen ajaa käynnistysvaihe, joka ei välttämättä vastaa todellisuutta simulaatiossa. Näin ollen käynnistysvaiheen tuloksia ei tavallisesti ole tarpeen ottaa huomioon lopullisten tulosten arvioinnissa.

Viimeinen vaihe on tulosten dokumentointi (Bangsow, 2020, s. 5). Dokumentaatiossa tulisi olla kuvaus suoritetuista työvaiheista ja yleiskatsaus tutkimuksen aikarungosta. Tutkimusraportin keskiössä tulee olla simulaatiotulokset esitettynä tutkimuksen kannalta kiinnostavista näkökulmista. Dokumentaatiossa voidaan myös ehdottaa jatkotoimenpiteitä järjestelmään liittyen.

2.4 Digitaalinen kaksonen

Digitaalinen kaksonen (engl. Digital Twin) on virtuaalinen jäljennös fyysisestä järjestelmästä, joka luodaan koostamalla ja yhdistämällä reaaliaikaista dataa useista lähteistä, kuten antureilta ja simulaatioista (Korhan, 2023, s. 19). Tällä virtuaalisella versiolla todellisesta vastineesta voidaan tehokkaasti jäljittää, tutkia ja toistaa sen toimintoja. Digitaalinen kaksonen parantaa ymmärrystä tarkasteltavasta kohteesta. Se tarjoaa ennakoivia näkemyksiä ja mahdollistaa perusteltujen päätösten tekemisen monilla osa-alueilla tarjoamalla peilikuvakopion todellisuudesta.

Molemmissa, simulaatiossa ja digitaalisessa kaksosessa, hyödynnetään digitaalisia malleja järjestelmän prosessien jäljentämiseen (IBM, 2021). Niiden ero on siinä, että simulaatiossa tarkastellaan usein yhtä tiettyä prosessia, kun taas digitaalinen kaksonen pystyy ajamaan useita hyödyllisiä simulaatioita tutkiakseen useita prosesseja. Lisäksi simulaatioissa ei yleensä hyödynnetä reaaliaikaista dataa, mutta digitaalinen kaksonen on suunniteltu kaksisuuntaiseen tiedonvälitykseen, jossa sensorit lähettävät dataa prosessorille ja se palauttaa luomansa oivallukset takaisin alkuperäiselle kohteelle. Paremmalla ja jatkuvasti päivittyvällä, monia osa-alueita käsittävällä datalla sekä virtuaalisen ympäristön laskentateholla digitaalinen kaksonen pystyy tutkimaan useampia kohteita, useammista näkökulmista tavallisiin simulaatioihin verrattuna.

2.5 Simulointiohjelmistot

Simulaatiokehitykseen tarkoitetut ohjelmistot voidaan jakaa kolmeen kategoriaan: yleiskäyttöisiin ohjelmointikieliin, kuten C, C++ tai Java, simulaatio-ohjelmointikieliin, kuten GPSS ja simulointiympäristöihin (Banks, 2005, s. 95–97). Simulointiympäristöillä on yhteisiä ominaisuuksia, kuten graafinen käyttöliittymä ja ne tukevat tavallisesti suurinta osaa simulaation osa-alueista. Useat simulointiympäristöt sisältävät työkaluja, jotka helpottavat statistiikan keräämistä ja analysointia. Niitä on saatavissa kaksi- ja kolmiulotteisilla sekä skemaattisilla grafiikoilla.

2.5.1 Visual Components

Vuonna 1999 perustettu Visual Components on suomalainen yritys, joka tuottaa simulaatio-ohjelmistoja (Visual Components, i.a.). Visual Components 4.0, joka on yrityksen uuden sukupolven simulaatioteknologiaa, julkaistiin vuonna 2016. Se sisältää versiot Essentials, Professional ja Premium. Visual Componentsin simulaatio-ohjelmistot mahdollistavat prosessien ja layoutien suunnittelun, optimoinnin sekä validoinnin.

2.5.2 Unity

Unity Technologiesin vuonna 2005 kehittämä Unity on yksi tämän hetken suosituimmista monialustaisista 2D- ja 3D-pelimoottoreista (Hatton, 2023; Zenva, 2024). Vaikka Unity on alun perin suunniteltu pelikehitykseen, sen käyttö on laajentunut merkittävästi myös muille aloille. Reaaliaikaisena pelimoottorina se sopii erinomaisesti digitaalisten kaksosten tietolähteitä hyödyntävien sovellusten kehitykseen (Unity Technologies, i.a.-a). Sen uusin versio, Unity 6 julkaistiin lokakuussa 2024 (Bromberg, 2024).

Teollisuudessa Unityn käyttökohteita (Unity Technologies, i.a.-b) ovat muun muassa

- työntekijöiden koulutus VR:n ja AR:n avulla
- virtuaalinen suunnittelu ja testaus
- suunnitelmien visualisointi
- simulointi.

3 VARASTONHALLINTAJÄRJESTELMÄ

3.1 Johdanto varastonhallintajärjestelmiin

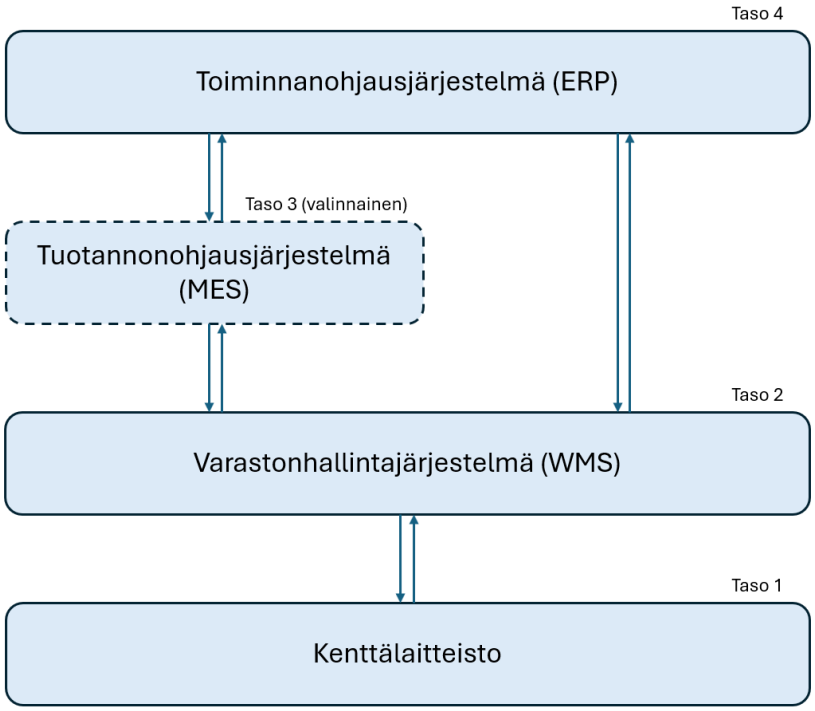
Varastolla tarkoitetaan tavaran väliaikaista säilytyspaikkaa, joka toimii puskurina toimitusketjulle (Richards, 2022, s. 1). Nykyisin varastot ovat avainasemassa varmistamassa, että asiakkaat saavat toimitukset ajallaan, kokonaisuudessaan ja vahingoittumattomana, jonka takia automaatioon ja teknologiaan investoidaan niissä yhä enemmän. Varastonhallintajärjestelmä on ohjelmisto, jonka avulla varaston toimintaa voidaan hallita ja valvoa (SAP, i.a.). Se voi olla itsenäinen järjestelmä tai osana toiminnanohjausjärjestelmää (Richards, 2022, s. 243).

WMS-järjestelmän tuomia mahdollisia etuja ovat (Richards, 2022, s. 244)

- reaaliaikainen varaston näkyvyys ja jäljitettävyys
- parempi tuottavuus
- paikkansapitävä varastokirjanpito
- väärinpoimintojen väheneminen
- automaattinen täydennys
- palautusten väheneminen
- tarkka raportointi
- parempi responsiivisuus
- etäyhteydellä saatavissa oleva tieto
- parempi asiakaspalvelu
- paperitöiden minimointi.

Tyypillisesti automaattivarastossa tuotannon suunnittelu, aikataulutus ja tilausten hallinta tapahtuu MES- tai ERP-järjestelmässä (Maunula, 2024). Tuotannonohjausjärjestelmä (engl. Manufacturing Execution System, MES) hallitsee ja valvoo tuotantoprosesseja reaaliajassa tehdastasolla, kun taas toiminnanohjausjärjestelmää (engl. Enterprise Resource Planning, ERP) käytetään yrityksen resurssienhallintaan (Werner, 2024). Järjestelmistä tieto kulkee WMS:lle, joka pitää kirjaa varaston inventaariosta ja allokoi tilaa tuleville tilauksille sekä materiaalia MES-järjestelmästä tulleisiin tilauksiin (Maunula, 2024).

Varastonohjausjärjestelmä (engl. Warehouse Control System, WCS) kommunikoi WMS-järjestelmän kanssa. Se ohjaa laitteita ja hallitsee materiaalivirtaa. Kuviossa 2 on esitetty järjestelmien välinen toiminta.



Kuvio 2. Tiedon kulkeminen ERP-, MES- ja WMS-järjestelmien välillä (Pesmel sisäinen tietolähde, 2025).

3.2 Pesmel WMS

Pesmel WMS -varastonhallintajärjestelmä on Pesmelin kehittämä järjestelmä, jonka ensimmäinen prototyyppi nykyisestä arkkitehtuurista on esitelty vuonna 2017 (Maunula, 2024). Siinä on neljä pääaluetta: integraatio asiakkaan tuotantojärjestelmiin, varastokirjanpidon ylläpito sekä materiaalivirtojen hallinta ja käyttöliittymät. Pesmel WMS yhdistää WMS:n ja WCS:n, mikä tekee siitä vakaan, mahdollistaa yksinkertaisemman käyttöliittymän ja vähentää synkronointiongelmista johtuvia käyttökatkoksia.

3.3 Käyttökokemus ja käyttöliittymä

Hyvin suunniteltu varastonhallintajärjestelmä parantaa tehokkuutta, vähentää kuluja ja johtaa korkeampaan asiakastyytyväisyyteen (Uitop, 2024). Suunnittelussa tulee ottaa huomioon käyttäjien tarpeet. WMS-käyttöliittymissä on suositeltavaa panostaa selkeyteen, navigoinnin helppouteen sekä johdonmukaisuuteen, jotta sen käyttö olisi mahdollisimman miellyttävää.

Käyttökokemus (engl. User Experience, UX) on yksi tärkeimmistä lähtökohdista digitaalisen tuotteen suunnittelussa (Ritter & Winterbottom, 2017, s. 1). Siihen vaikuttaa laaja joukko toisiinsa liittyviä tekijöitä, mutta sen ytimessä on käytettävyys (mts. 8). Tuotteen tai palvelun voidaan sanoa olevan käytettävä, kun käyttäjä kykenee suorittamaan haluamansa tehtävän odottamallaan tavalla ilman ongelmia (Rubin & Chisnell, 2008, s. 4).

Käyttöliittymä (engl. User Interface, UI) on käyttäjän ja laitteen välinen interaktiivinen kerros (Ritter & Winterbottom, 2017, s. 70–71). Hyvin suunniteltuna se parantaa käytettävyyttä ja saavutettavuutta sekä kohentaa käyttökokemusta kokonaisvaltaisesti. Käyttöliittymäsuunnittelu on äärimmäisen tärkeää käyttäjän kannalta (Mátrai, 2010, s. 7). Hyvin suunniteltu käyttöliittymä helpottaa käyttäjää toistuvien tehtävien suorittamisessa ja ehkäisee ongelmilta. Käyttöliittymäsuunnitteluun on olemassa monia suuntaviivoja, mutta teknologian kehittyessä uusia haasteita ilmaantuu jatkuvasti.

3.4 Graafinen käyttöliittymä

Graafinen käyttöliittymä (engl. Graphical User Interface, GUI) on nykyään tietokoneissa yleisimmin käytetty käyttöliittymätyyppi, jota mm. Windows-käyttöjärjestelmä hyödyntää (Levy, 2024). Se on korvannut aiemmat tekstipohjaiset käyttöliittymät helpommin opittavalla ja käyttökokemukseltaan paremmalla järjestelmällä. Graafisessa käyttöliittymässä interaktio tapahtuu visuaalisia elementtejä, kuten kuvakkeita ja painikkeita klikkaamalla (Rosencrance, 2024). Se on helppokäyttöinen ja nopeasti opittava, ja sen ulkoasun tulisi olla selkeä ja yhdenmukainen sekä nopea ja responsiivinen hyvän käyttökokemuksen varmistamiseksi.

4 NYKYINEN SIMULOINTI- JA KÄYTTÖLIITTYMÄKEHITYSPROSESSI

4.1 Myyntisimulointi

Myyntiprosessi alkaa myytävän järjestelmän 2D-layoutin tekemisellä. Tämän pohjalta toteutetaan myyntiprojektikohtaisesti 3D-layout joko 3D-mallinnusohjelmalla tai Visual Componentsilla. 3D-mallinnusohjelmalla tehtyjä layouteja on mahdollista käyttää myyntitarkoitukseen sellaisenaan, kun taas Visual Componentsilla 3D-layoutiin saadaan mallinnettua myös liikettä. Laitteiden liikkeitä voidaan mallintaa visualisointitarkoituksessa, tai simulointia varten.

Visual Components -myyntisimulaatiot voidaan jakaa kahteen kategoriaan, visualisaatioihin ja simulaatioihin. Myyntivisualisaatioilla on tarkoitus havainnollistaa myytävän järjestelmän toimintaa asiakkaalle, jolloin niissä keskitytään enemmän visuaalisuuteen, eikä niillä kerätä dataa järjestelmästä simulaation tavoin. Myyntisimulaatioilla simuloidaan järjestelmän toimintaa ja kerätään simulaatioajoilta statistiikkaa, kuten läpimenoaikoja. Molemmat ovat tärkeä osa myyntivaihetta ja useissa tapauksissa tehdään sekä simulaatio että visualisaatio. Niiden avulla pystytään varmentamaan myytävien järjestelmien toiminta sekä havainnollistamaan niiden toimintaperiaate asiakkaille.

Myyntisimulaation tekeminen Visual Components -ohjelmistolla alkaa tavanomaisesti varastohyllyn mallinnuksella, jonka jälkeen valitaan loput tarvittavat komponentit. Visual Componentsilla on mahdollista mallintaa omia komponentteja alusta alkaen tai tuoda CAD-malleja simulointiympäristöön. Valmiit komponentit voidaan tallentaa, jolloin niitä on mahdollista hyödyntää myös muissa simulaatioissa. Komponenttien toiminnallisuus ohjelmoidaan Visual Componentsin Python APIn avulla, jolla niihin saadaan mukautettuja ominaisuuksia sekä logiikkaa (Visual Components, 2017a).

Valmiista simulaatiosta kerätään tarvittava statistiikka. Visual Components sisältää statistiikkatyökalun, jonka avulla voidaan kerätä haluttua dataa tietyn tai tiettyjen komponenttien osalta (Visual Components, 2017b). Simulaatioajon aikana statistiikka piirtyy halutun tyyliseen kaavioon, kuten piirakka- tai viivadiagrammiin Statistics-välilehdelle. Komponentti voidaan mallintaa myös siten, että sillä on yksi tai useampi ennalta määritelty tila, joita

hallitaan Python APIn avulla (Visual Components, 2017c). Tilat määritellään Visual Componentsin statistiikkatoiminnolla ja niitä voidaan käyttää hallitsemaan muiden ominaisuuksien toiminnallisuutta. Esimerkiksi laitteiden käyttöasteet saadaan tilastoitua määrittelemällä niille tilat, kuten Busy, Idle ja Blocked Python APIssa ja esittämällä ne piirakkadiagrammissa. Kerätty statistiikka on mahdollista viedä Visual Componentsista Exceliin.

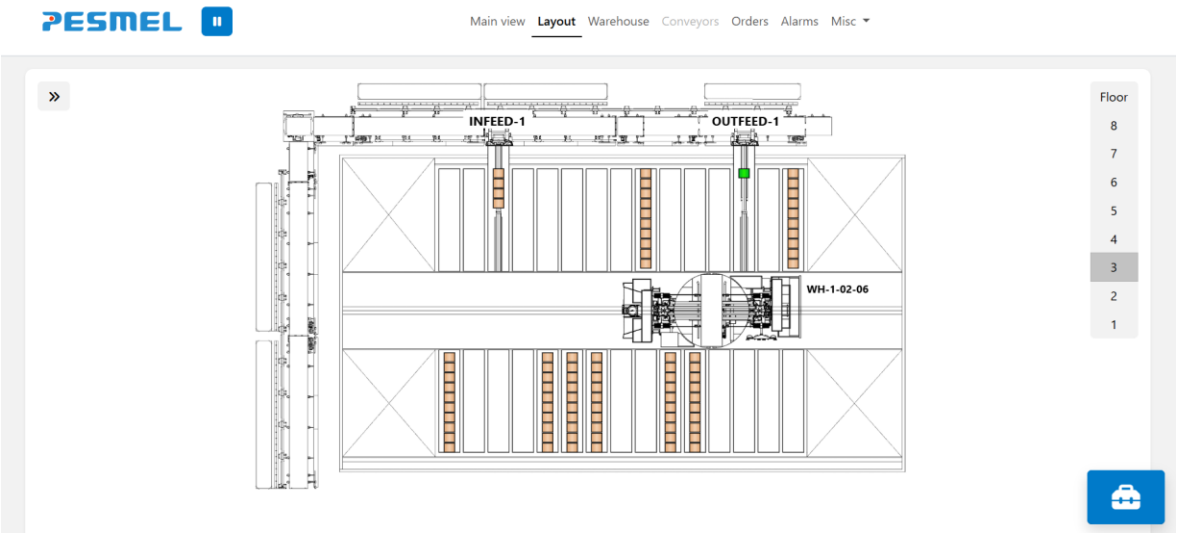
Lopuksi simulaatiomallista tehdään video, josta selviää järjestelmän keskeinen toiminta. Visual Components Experience on ohjelma, jonka avulla Visual Components -simulaatiot voidaan tallentaa 3D-animaatioiksi .vcax-muodossa (Visual Components, 2024). Tämän formaatin tiedostoja voidaan avata vain Visual Components Experience -ohjelmalla.

4.2 WMS-käyttöliittymä

Selainpohjainen WMS-käyttöliittymä koostuu useista sivuista, joiden kautta varastoa ja siihen liittyvää tietoa hallitaan ja valvotaan. Oleellisimpia ovat layout-, warehouse-, hälytykset- ja tilaukset-sivut (Maunula, 2024). Näiden lisäksi käyttöliittymä sisältää sivut mm. raporteille, tietokannalle ja parametreille. WMS-käyttöliittymissä on tietyt vakio-ominaisuudet, mutta joissain tapauksissa niitä voidaan kustomoida asiakkaan tarpeiden mukaisesti.

4.2.1 Layout-sivu

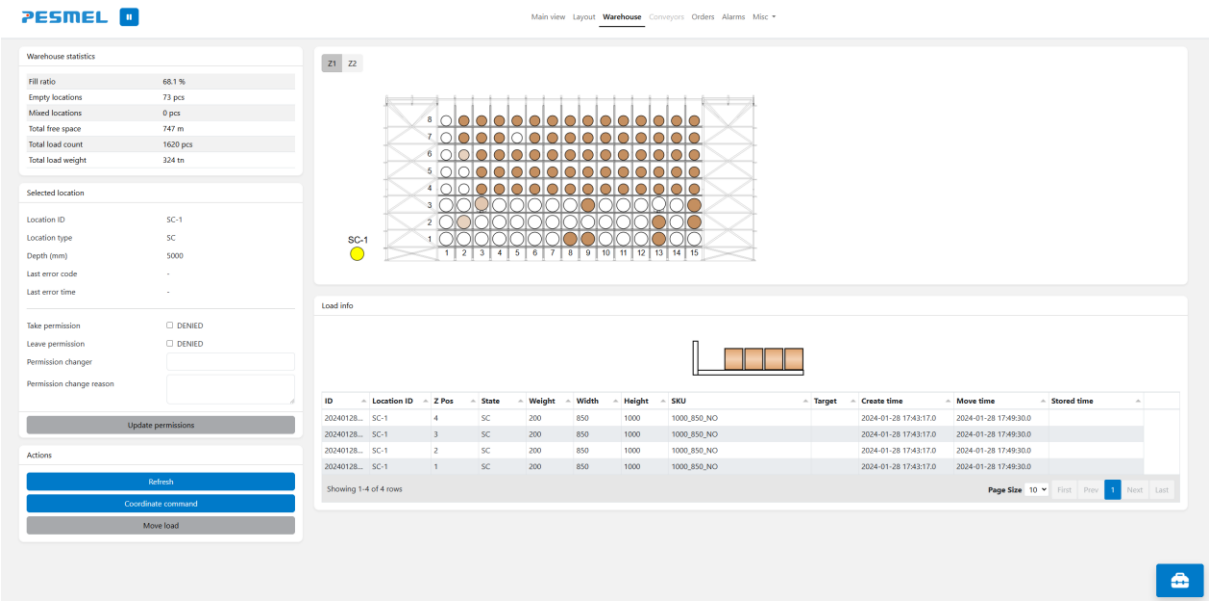
Layout-sivulla esitetään varaston pohjakuva sekä yleiskatsaus sen reaaliaikaisista tapahtumista (Maunula, 2024). Jos varastossa tapahtuu jotain poikkeavaa, kuten laitteen pysähtyminen, sitä indikoidaan punaisella värillä, joka kiinnittää operaattorin huomion. Sivulla on kerrosvalinta, josta voidaan valita haluttu varaston kerros tarkasteluun. Varastossa olevaa rullaa klikkaamalla avautuu ponnahdusikkuna, joka sisältää perustietoa rullasta, kuten sen id-numeron ja varastopaikan. Tarvittaessa laitteita on mahdollista pysäyttää jossain määrin sivulta käsin. Kuvassa 2 on WMS-käyttöliittymän layout-sivu.



Kuva 2. WMS-käyttöliittymän layout-sivu, jossa varaston 2D-kuva ylhäältä kuvattuna.

4.2.2 Warehouse-sivu

Warehouse-sivulla esitetään varaston leikkauskuva sivusta päin (Maunula, 2024). Varastokanavien tilanne voidaan tarkistaa niitä klikkaamalla. Jos kanava on jostain syystä kiellossa, sitä indikoidaan punaisella värillä. Lisäksi sivulla on esitettynä varaston statistiikkaa ja yksittäisen varastokanavan tietoja. Kuvassa 3 on WMS-käyttöliittymän warehouse-sivu.



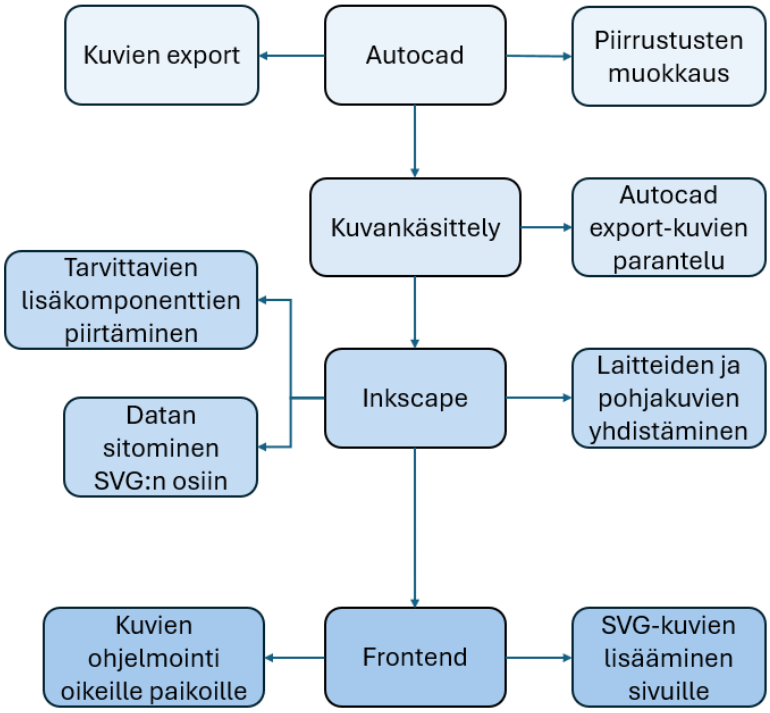
Kuva 3. WMS-käyttöliittymän warehouse-sivu, jossa varaston 2D-kuva sivultapäin kuvattuna.

4.3 Kaksiulotteisen WMS-layoutin kehittäminen

WMS-käyttöliittymien pohja- ja leikkauskuvat toteutetaan kaksiulotteisina SVG-muodossa (Ämmälä, 2024). Prosessi alkaa varaston kaksiulotteisen layoutin CAD-mallin yksinkertaistamisella muokkaamalla siitä epäolennaiset osat pois. Sen jälkeen Autocadista viedään tarvittavat pohjakuvat eri kerroksille, sivukuva eli warehouse view ja laitteiden kuvat. Autocadista vietyjä kuvia on tarpeen parannella kuvankäsittelyllä, jotta niistä saadaan tarvittavan selkeitä.

Käsitellyt kuvat tuodaan Inkscape-ohjelmaan, joka on avoimen lähdekoodin editointiohjelma vektorigrafiikoille (Ämmälä, 2024; Inkscape, i.a.). Vektorigrafiikka mahdollistaa tarkat kuvat rajoittamattomalla resoluutiolla (Inkscape, i.a.). Inkscape käyttää pääasiassa SVG-tiedostomuotoa, joka on tuettu monissa sovelluksissa, mukaan lukien verkkoselaimissa.

Inkscapessa kuvankäsittelyllä paranneltuihin Autocadista tuotuihin kuviin piirretään tarvittavat lisäkomponentit, kuten turva-alueet ja ovet sekä yhdistetään laitteiden kuvat pohjakuviin (Ämmälä, 2024). Sen lisäksi data, kuten laitteiden liikkeet, sidotaan SVG:n osiin. Tämän jälkeen SVG-kuvat viedään frontendiin, jossa ne lisätään web-sivuille ja ohjelmoidaan näkymään oikeilla paikoilla. Kuviossa 3 havainnollistetaan WMS SVG -layoutin kehittämisen vaiheita.



Kuvio 3. SVG-layoutin kehittämisen vaiheet (Ämmälä, 2024).

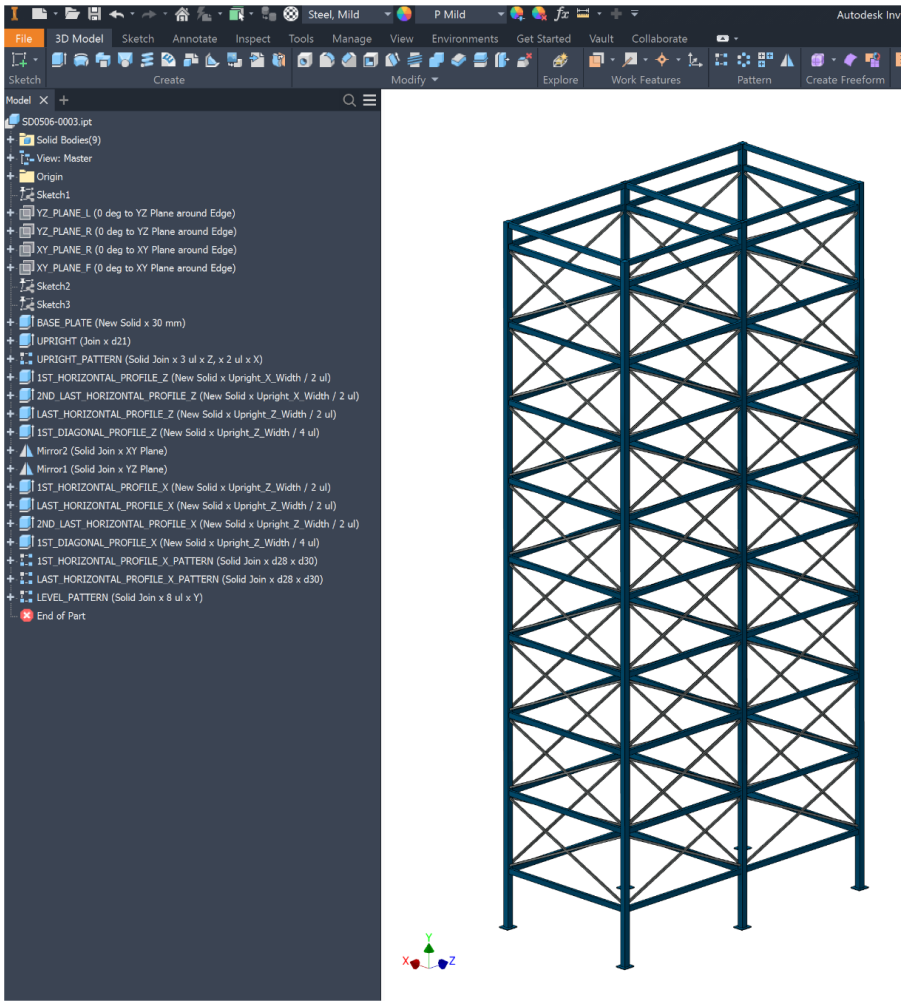
4.4 WMS-simulointi

WMS-simulointia varten on kehitetty ohjelma mallintamaan tilannetta, jossa WMS kommunikoi kentällä sijaitsevien laitteiden kanssa (Maunula, 2024). Ohjelma sisältää kaikki laitteiden liikkeet reaaliajassa. WMS-simulointi mahdollistaa järjestelmien validoinnin, optimoinnin, auttaa ennakoimaan tilanteita ja helpottaa käyttöönottoa.

5 UUSI SIMULOINTI- JA KÄYTTÖLIITTYMÄKEHITYSPROSESSI

5.1 3D-mallinnus

Prosessi alkaa varaston mallinnuksella. Varastomallin tekeminen voidaan aloittaa täysin tyhjästä tai olemassa olevaa mallia muokkaamalla (T. Mäki-Jussila, henkilökohtainen tiedonanto, 9.12.2024). Kokonaan uuden mallin tekeminen aloitetaan piirtämällä perusmuoto, johon lisätään geometrisia rajoitteita ja parametreihin sidottuja mittoja, jolloin mallista saadaan helposti muokattava. Tämän jälkeen siihen lisätään geometria, esimerkiksi pursottamalla. Geometriaan voidaan lisätä yksityiskohtia, kuten reikiä, viisteitä tai pyöristyksiä. Kuvassa 4 on esitetty edellä kuvatun prosessin lopputulos, jossa perusmuotoja on lisäksi peilattu ja monistettu.



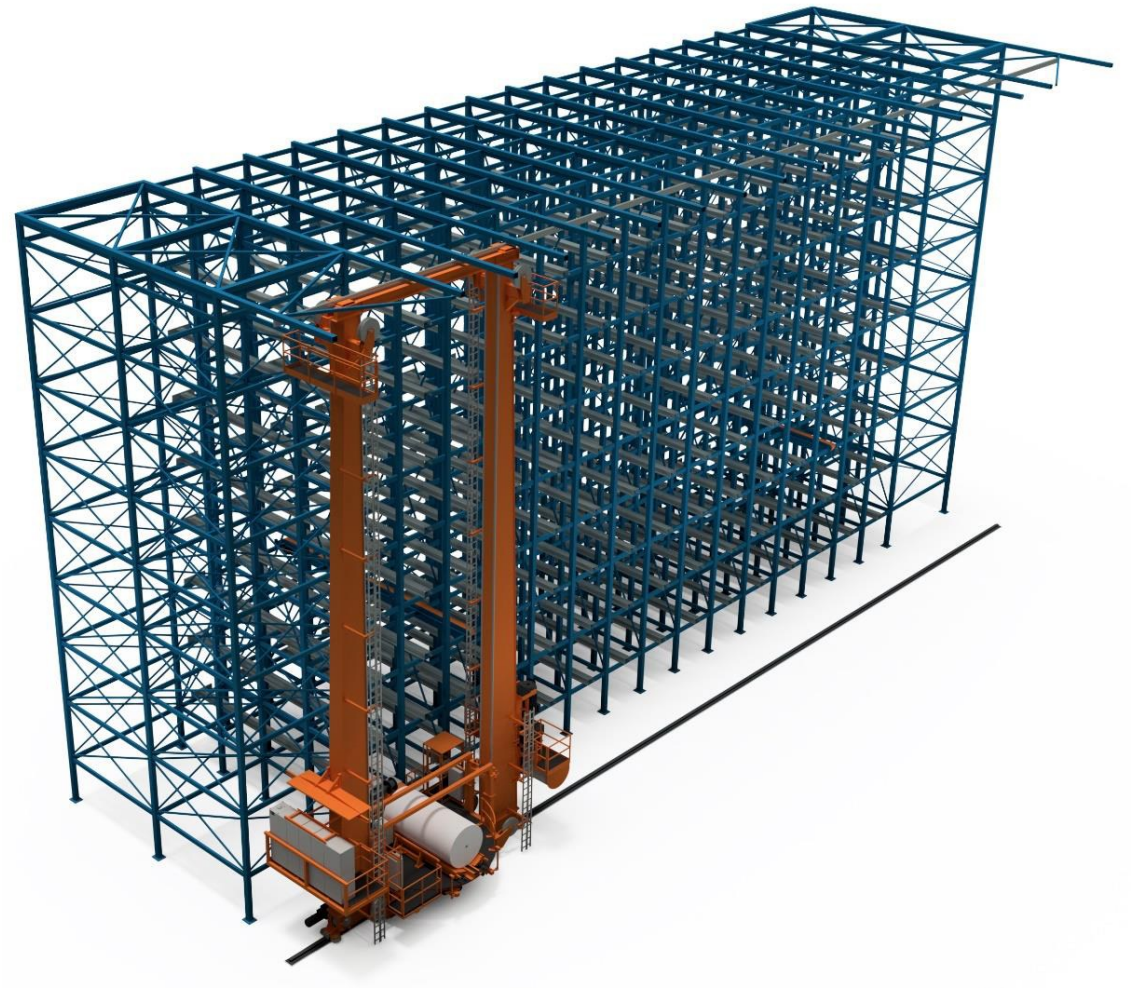
Kuva 4. Yksinkertaistettu 3D-malli varastohyllyn päätyrakenteesta (Mäki-Jussila, 2024a).

Kun muokataan olemassa olevaa mallia, etsitään ensin käyttötarkoitukseen sopiva pohjamalli (T. Mäki-Jussila, henkilökohtainen tiedonanto, 9.12.2024). Malli muokataan kevennetyksi versioksi, jolloin siitä poistetaan pienkomponentit, kuten pultit, mutterit ja osa mallissa olevista rei'istä. Lisäksi näkymättömät ontelot pursotetaan täyteen. Mallista tehdään parametrinen, jotta sen kokoa voidaan helposti muuttaa. Kuvassa 5 on esimerkki hyllystöhissin alkuperäisestä ja muokatusta 3D-mallista.

Kuva 5. Hyllystöhissin alkuperäinen (vas.) ja muokattu 3D-malli (Mäki-Jussila, 2024b).

Lopuksi tehdään pääkokoonpano, joka sisältää kaikki mallinnetut komponentit ja kokonaisuudet (T. Mäki-Jussila, henkilökohtainen tiedonanto, 9.12.2024). Pääkokoonpanossa

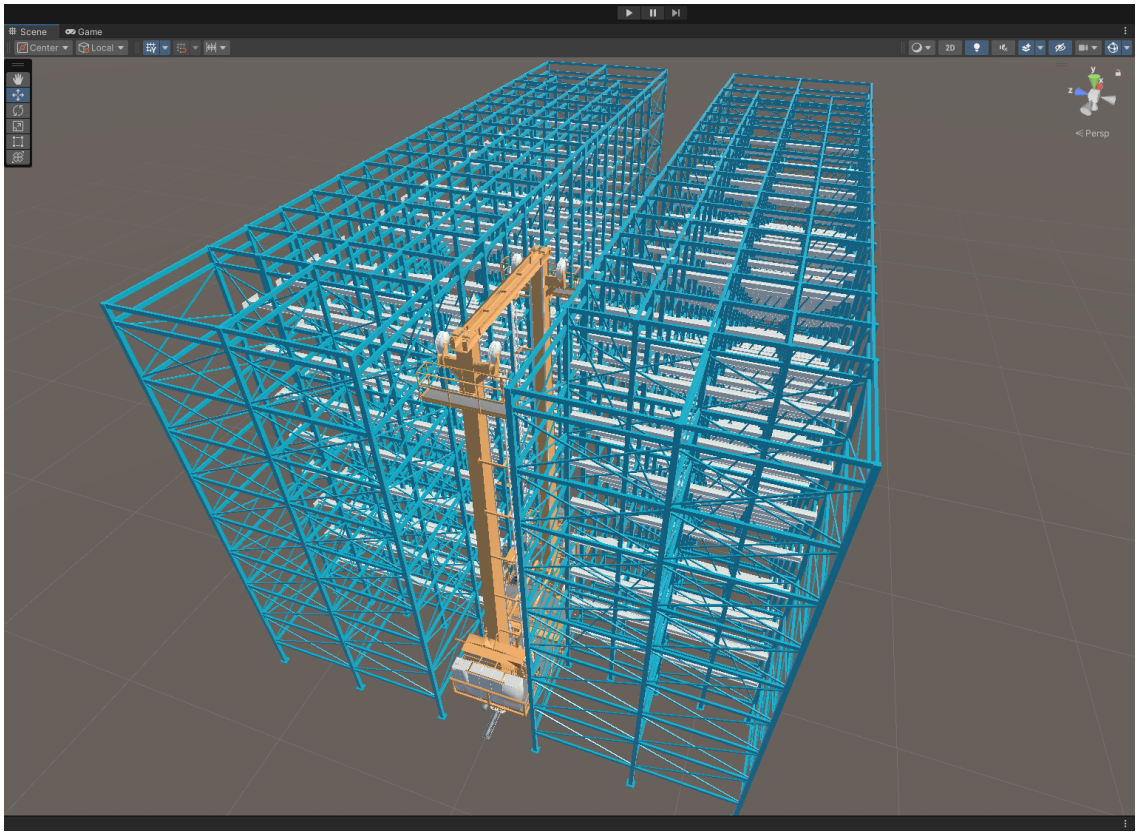
osille määritetään tarvittavat rajoitteet. Kuvassa 6 on esimerkki puolileikatusta pääkokoonpanosta.



Kuva 6. Esimerkki pääkokoonpanosta (Mäki-Jussila, 2024c).

5.2 Varastomallin tuominen Unity-ympäristöön

Kun varaston 3D-malli on valmis, voidaan se tuoda Unity-ympäristöön. 3D-mallien tallentaminen yleisimmistä mallinnusohjelmista on mahdollista niiden alkuperäisessä tiedostomuodossa, kuten .fbx, .max, .blend, .mb ja .ma (Unity Technologies, i.a.-c). Tässä tilanteessa Unity kutsuu 3D-mallinnusohjelman fbx-vientilisäosaa tuodakseen tiedoston ympäristöön. Tässä työssä malli tuotiin Unityyn .fbx- ja .blend-tiedostomuodoissa. Molemmissa tapauksissa mallin tuonti tapahtui raahaamalla tiedosto ympäristöön. Kuvassa 7 on korkeavaraston 3D-malli Unity-ympäristössä.



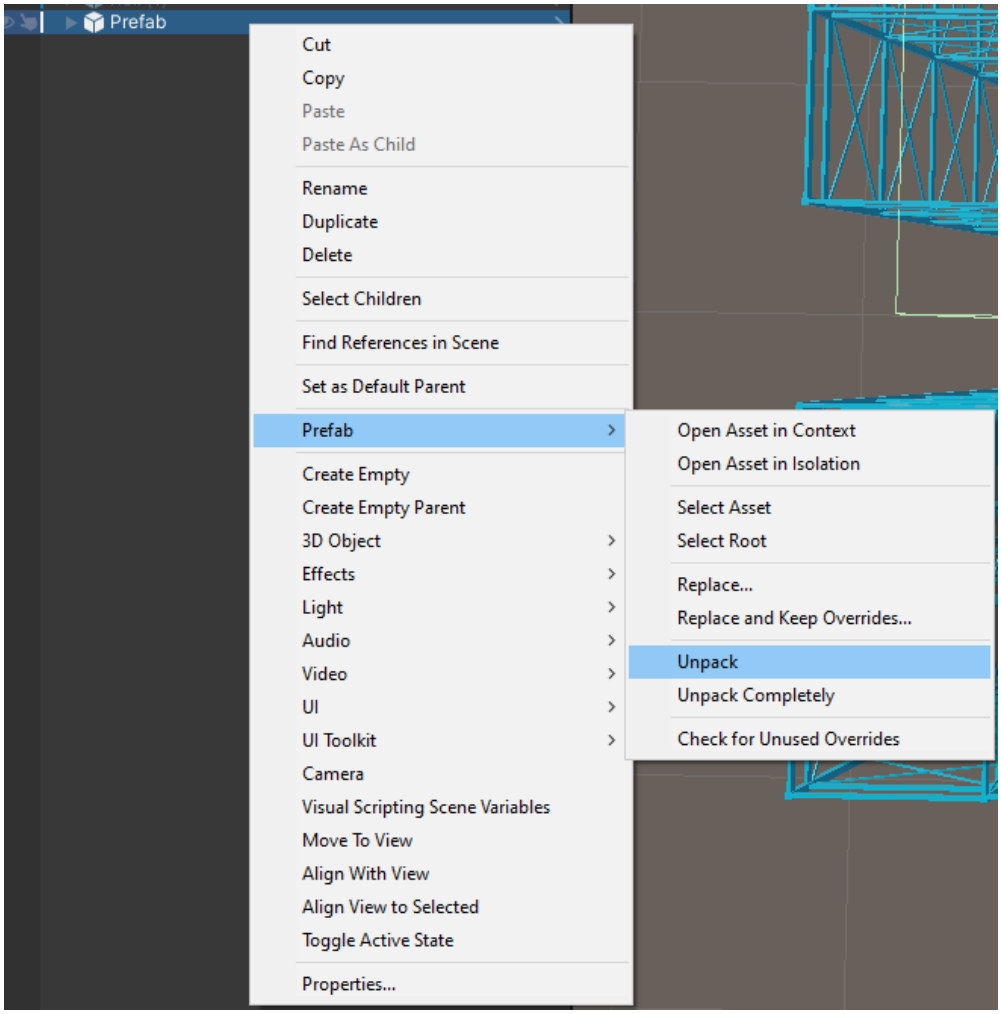
Kuva 7. Varastomalli Unity-ympäristössä.

Unityssä on vasenkätinen koordinaatisto, jossa y-akseli osoittaa ylöspäin (Unity Technologies, i.a.-d). Kun malleja tuodaan ympäristöön, on hyvä ottaa huomioon, etteivät 3D-mallinnusohjelmien ja Unityn koordinaatistot välttämättä vastaa toisiaan. Esimerkiksi työssä käytetty 3D-malli on mallinnettu Autocad Inventor -ohjelmalla, jossa on oikeakätinen koordinaatisto, z-akselin ollessa pystyakseli (Ekins, 2008, s. 17). Varastomalli piti kääntää Unityn z- ja y-akselien suhteen 90 astetta, jotta niiden koordinaatit saatiin vastaamaan toisiaan.

Unity-käytössä 3D-mallien tulee olla kevennettyjä, sillä liian yksityiskohtainen malli on tarpeettoman raskas. Huomioitavaa on myös 3D-mallin skaalaus Unityssä, jossa sen fysiikka- ja valaistusjärjestelmät olettavat yhden metrin Unity-ympäristössä vastaavan yhtä unitia ympäristöön tuodussa 3D-mallissa (Unity Technologies, i.a.-e). Esimerkiksi .fbx-tiedoston skaalauskerroin on 0,01.

5.3 Mallin pilkkominen osiin

Kun varastomalli on tuotu Unityyn, se täytyy purkaa. Purkamisen seurauksena mallin osat muuttuvat erillisiksi GameObjecteiksi, jolloin niitä on mahdollista muokata osakohtaisesti (Unity Technologies, 2018). Se tapahtuu klikkaamalla mallia hierarkiavalikossa hiiren oikealla painikkeella ja valitsemalla ”Unpack Prefab”. Purkamisen seurauksena luotu GameObject Unityn scene-ympäristössä ei ole enää linkitetty alkuperäiseen prefab-assetiin. Kuvassa 8 on esitettynä vaiheet mallin purkamiseen.



Kuva 8. Prefabin purkaminen Unityssä.

Purkamisen jälkeen GameObjectien välille luodaan tarvittavat parent-child-suhteet. Child-objekti liikkuu, kääntyy ja skaalautuu parent-objektin mukaisesti (Unity Technologies, 2020). Objektilla voi olla useita childeja, mutta vain yksi parent. Lukuisat parent-child-tasot

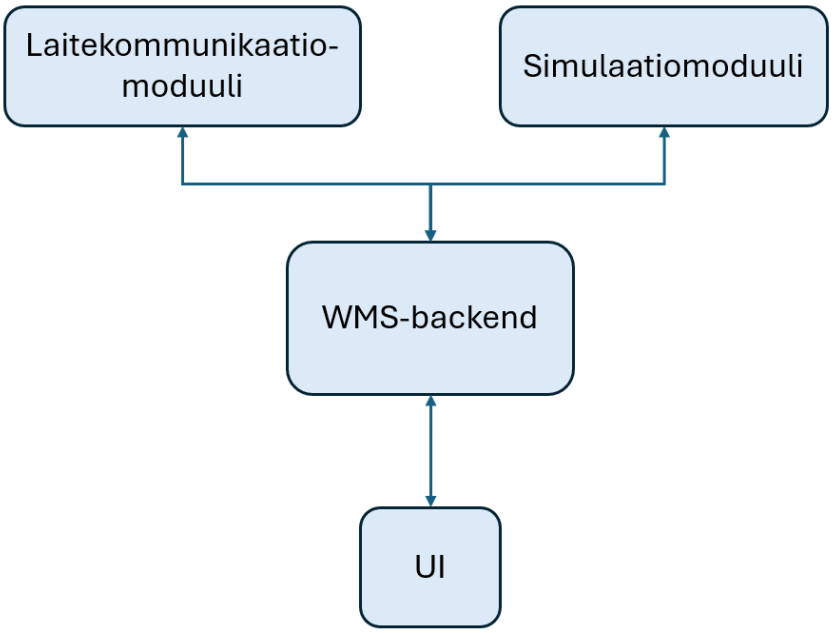
muodostavat hierarkian, jossa ylin GameObject, jolla ei ole parentia, on sen juuri. Parent-objekti voidaan luoda vetämällä GameObject toiseen. Samalla näiden GameObjectien välille muodostuu parent-child-suhde.

5.4 Myyntivisualisaatio

Unityllä kehitettiin järjestelmän toimintaa havainnollistava myyntivisualisaatio, jossa laitteiden toiminnallisuus toteutettiin animoimalla niiden liikkeet C#-ohjelmointiskriptien avulla. Siinä hissille luotiin tehtäviä, joiden perusteella se haki tai jätti rullan varastokanavaan tiettyyn ennalta määrättyyn lokaatioon. Rullat, jotka hissin oli määrä hakea, ohjelmoitiin näkymään hakupaikoille. Niiden lisäksi varastoon luotiin rullia tiettyihin kohtiin kasvattamaan sen täyttöastetta. Visualisaatiossa tuotiin ilmi järjestelmän keskeinen toimintaperiaate, johon kuului rullien siirtely, hakeminen sisäänsyötöstä varastoon ja sieltä ulossyöttöön.

5.5 WMS-integraatio

Tiedon hakeminen WMS-backendilta. Unityn ja WMS-backendin väliseen tiedonsiirtoon käytetään REST-rajapintaa ja WebSocket-protokollaa. WebSocket mahdollistaa kaksisuuntaisen kommunikaation asiakkaan ja palvelimen välillä (Diaconu, 2024). Backend saa laitteiden sijainti- ja tehtävätiedon sekä tiedon varastossa olevista rullista laitekommunikaatio- tai simulaatiomoduulilta, tässä tapauksessa jälkimmäiseltä. Hissin liikkeet ohjelmoitiin lisäämällä C#-ohjelmointiskripti jokaiselle sen liikkuvalle osalle. Ohjelmakoodeissa luetaan WebSocketilta tulevaa dataa, joka määrittää hissin liikkeet x-, y- ja z-suunnissa reaaliajassa. Varastossa olevien rullien paikkatieto haetaan tietyin aikavälein REST-rajapinnan avulla. Se lähettää HTTP-pyynnön palvelimelle, joka palauttaa vastauksen JSON-muodossa (Red Hat, 2020). Rullia visualisoitiin luomalla rulla-prefab varastohyllyyn backendilta saadun rullaposition kohdalle. Kuviossa 4 havainnollistetaan tiedon kulkemista moduuleilta WMS-backendille ja sieltä WMS-käyttöliittymälle.



Kuvio 4. Tiedon kulkeminen.

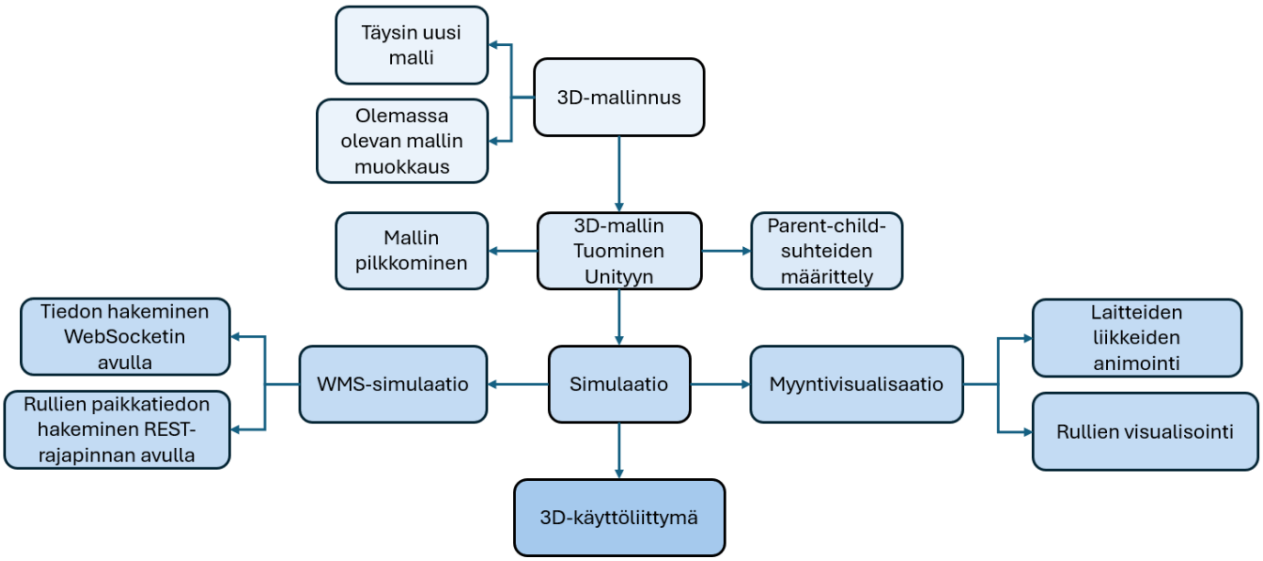
WMS 3D -käyttöliittymä. Tässä työssä käyttöliittymän kehittäminen ei ollut keskiössä, mutta Unity-malliin lisättiin mahdollisuus kuvakulman säätöön sekä syöttökenttä tokenin lisäämistä helpottamaan. Kuvakulman säätö toteutettiin asettamalla Unity-sceneen kameroita eri kuvakulmiin ja vaihtelemalla aktiivista näkymää niiden välillä UI Button -komponentin avulla. Kameroita oli kolme, joista yksi kuvasi varastoa suoraan sivulta, toinen suoraan ylhäältä ja kolmas oli kolmiulotteinen näkymä. Kolmiulotteiseen näkymään sisältyi vapaalentotila, jonka avulla varastossa pystyi liikkumaan ikään kuin lentäen näppäimistön w-, a-, s- ja d-näppäimiä käyttäen.

Backend-yhteys autentikoidaan selaimelta saatavalla tokenilla, jonka on oltava ajan tasalla, jotta Unity-simulaatio toimii. Käyttöliittymään lisättiin UI Input Field -komponentti, joka helpottaa tokenin määrittämistä. Syöttökenttä ohjelmoitiin siten, että siihen syötetty teksti määrittää uuden tokenin.

WebGL-upotus WMS-selainversioon. Unityn WebGL-rakenteen luominen tapahtuu kääntämällä ohjelmasta WebGL-versio Unityssä (V. Pihlajamäki, henkilökohtainen tiedonanto, 11.2.2024). Kääntäminen luo tarvittavat tiedostot selainpohjaista suorittamista

varten. Seuraavaksi Unityn WebGL upotetaan Aurelia-sovellukseen. Aurelia-sovelluksessa WebGL-käännös upotetaan HTML-sivulle lisäämällä se dynaamisesti ladattavana osana canvas-elementtiin. Tämä tehdään luomalla erillinen komponentti, joka lataa WebGL-sovelluksen ja sijoittaa sen sivulle sopivaan kohtaan.

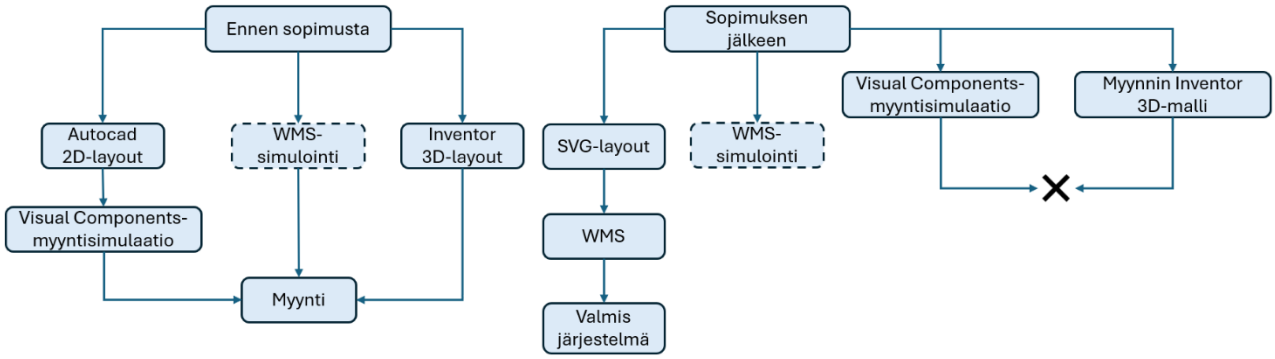
Unityn avulla WMS-simulaatioista saadaan kolmiulotteisia, aiemman kaksiulotteisen sijaan. Ne toimivat kuitenkin samalla periaatteella reaaliajassa laitekommunikaatio- tai simulaatiomoduulin tiedon perusteella. Kolmiulotteisuuden ansiosta simulaatioita voidaan tarkastella useammista kuvakulmista, mikä tekee järjestelmäkokonaisuuksien hahmottamisesta helpompaa. Lisäksi kolmiulotteiset simulaatiot näyttävät realistisemmilta kaksiulotteisiin verrattuna. Kuviossa 5 on havainnollistettuna uuden simulaatio- ja käyttöliittymäkehitysprosessin vaiheet kokonaisuudessaan.



Kuvio 5. Uuden simulaatio- ja käyttöliittymäkehitysprosessin vaiheet.

6 TOIMINTAMALLIEN VERTAILU

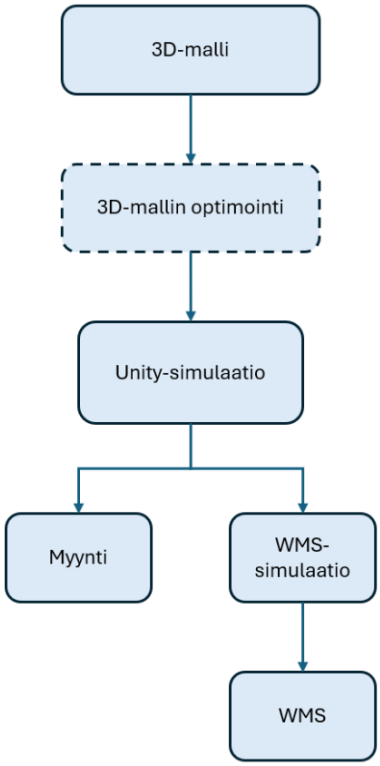
Nykyisessä toimintamallissa voidaan sanoa olevan kaksi vaihetta kuvion 6 mukaisesti: ennen sopimusta ja sen jälkeen. Myyntiprosessi lähtee liikkeelle Autocad 2D- tai Inventor 3D -layoutista. 2D-layoutin perusteella tehdään Visual Components -myyntisimulaatio tai -visualisaatio, kun taas 3D-layoutia voidaan käyttää suoraan myyntitarkoitukseen. Joissain tapauksissa kuitenkin toteutetaan molemmat osat.



Kuvio 6. Nykyinen toimintamalli.

Kun sopimus asiakkaan kanssa on tehty, alkaa kuvion 3 mukainen SVG-layoutin kehittämisvaihe, josta prosessin edetessä muodostuu valmis WMS. Myynnin simulaatio- tai visualisaatio ja 3D-malli jäävät tarpeettomiksi, koska niitä ei voida hyödyntää WMS-kehityksessä. WMS-simulointia tehdään tapauksen mukaan joko nykyisen toimintamallin molemmissa tai vain toisessa vaiheessa.

Tavoitteena on, että toimintamallia muuttamalla voitaisiin siirtyä kokonaisuudessaan suoraviivaisempaan prosessiin. Uusi toimintamalli lähtee liikkeelle 3D-mallista, joka optimoidaan tarpeen mukaan. Mallin pohjalta tehdään Unityssä simulaatio tai visualisaatio myynnin käyttöön. Myyntivaiheen jälkeen tätä samaa Unity-projektia voidaan käyttää WMS-kehitykseen. Kuviossa 7 on esitetty uuden toimintamallin vaiheet.



Kuvio 7. Uusi toimintamalli.

Nykyisessä toimintamallissa SVG-layoutien tekeminen käsittää monta vaihetta, minkä vuoksi se on työlästä, kallista ja hidasta. Unityyn 3D-malli voidaan tuoda suoraan käyttökelpoisessa muodossa raahaamalla se ympäristöön. Monimutkaisia malleja tulee kuitenkin yksinkertaistaa, jotta ne eivät ole liian raskaita käyttötarkoitukseen nähden, jolloin prosessiin tulee yksi työvaihe lisää. Kun Unity-projektista on olemassa toimiva malli, uusien WMS-simulaatioiden tekeminen helpottuu, koska samoja ohjelmakoodeja voidaan hyödyntää eri tapauksissa. Koodeissa on peruselementit Unity-mallin ja WMS-backendin yhdistämiseen sekä laitteiden liikkeisiin, jolloin kaikkea ei tarvitse ohjelmoida uudelleen jokaisen projektin kohdalla.

Myyntisimulaatioiden kannalta Visual Components on yksinkertaisempi työkalu Unityyn verrattuna, sillä se on varta vasten simulointikehitykseen suunniteltu ohjelmisto ja sisältää valmiita työkaluja mm. simulaatioiden validointiin ja optimointiin sekä jossain määrin komponenttien mallinnukseen. Lisäksi yrityksessä on jo ennalta osaamista Visual Componentsilla simuloimisesta. Toisaalta Unity on kokonaisuudessaan monipuolisempi työkalu. Siinä

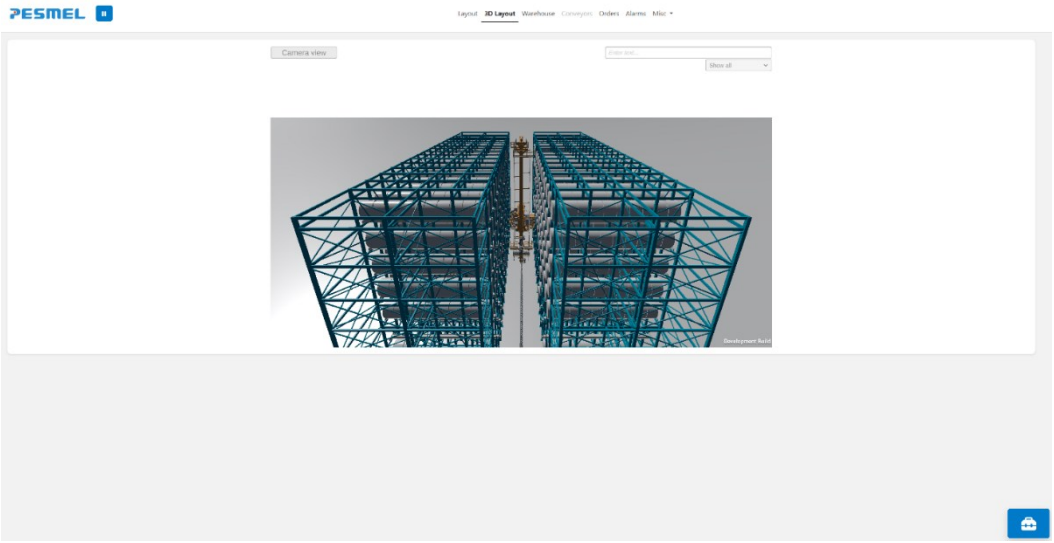
asioiden toteuttamiseen vaaditaan kuitenkin enemmän manuaalista työtä. Tämän takia uutta toimintamallia käyttöönotettaessa tulisi erityisesti alkuvaiheen tavoitteena olla uudelleenkäytettävien kirjastojen rakentaminen, jolloin työn määrä vähenisi tulevaisuudessa. Tavoitteena oli, että Unityllä tehtävissä myyntisimulaatioissa hyödynnettäisiin WMS-backendilta saatavaa dataa, jolloin WMS-simulaatioiden ohjelmakoodeja voitaisiin hyödyntää myös myyntisimulaatioissa ja toisin päin. Tässä työssä kuitenkin toteutettiin myyntivisualisaatio animoimalla laitteiden liikkeet erillisillä C#-skripteillä.

Nykyisten myyntivaiheen simulaatioiden ja visualisaatioiden jakaminen yrityksessä sisäisesti ja asiakkaille on jokseenkin haastavaa, sillä valmiiden mallien avaamiseen vaaditaan Visual Components -ohjelmisto. 3D-animaatioiden kohdalla on sama ongelma, sillä Experience-animaatiot voidaan avata vain Visual Components Experience -ohjelmalla. Unityn tuoma etu on, että sen WebGL-ominaisuuden avulla simulaatiot ja visualisaatiot voitaisiin julkaista verkkosivulle, jolloin niiden jakaminen onnistuu kyseisen sivun linkillä ja niitä voidaan katsoa selaimen välityksellä.

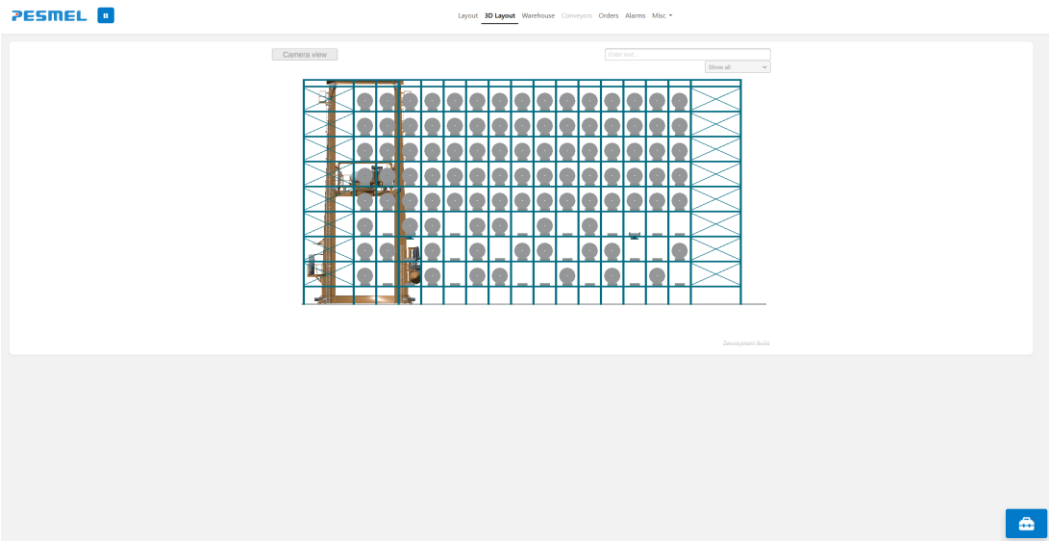
Kun tarkastellaan kokonaisuutta, Unityllä toteutettava simulointi- ja käyttöliittymäkehitysprosessi on aiempaa suoraviivaisempi ja tehokkaampi toimintatapa. Nykyisen toimintatavan huono puoli on sen monivaiheisuus ja jakautuminen selkeästi kahteen erilliseen vaiheeseen. Tällöin tuloksena on se, että myyntivaiheessa tehtyjä simulaatioita, visualisaatioita ja malleja ei voida hyödyntää WMS-kehityksessä, koska sopimuksen jälkeisessä vaiheessa SVG-layoutin tekeminen käynnistyy nollapisteestä. Uuden toimintatavan myötä myynti- ja WMS-kehitysvaiheet olisivat molemmat mahdollista toteuttaa Unity-ympäristössä yhden 3D-mallin pohjalta. Samalla käytettyjen työkalujen määrä eri vaiheissa saataisiin karsittua optimaalisimmillaan kahteen: 3D-mallinnusohjelmaan ja Unityyn. Uuden toimintamallin käyttöönotto vaatii kuitenkin merkittäviä muutoksia nykyisiin toimintatapoihin, ja siihen tulee olla selkeä suunnitelma. Sen myötä prosessia olisi kuitenkin mahdollista tehostaa, jolloin muutosten tekeminen on kannattavaa.

7 TULOKSET JA POHDINTA

Työn tuloksena kehitettiin myyntivisualisaatio ja WMS-simulaatio Unity-ympäristössä. WMS 3D -layout saatiin upotettua WMS-selainversioon Unityn WebGL-ominaisuutta hyödyntäen, kuten kuvissa 9 ja 10 esitetään. WMS-simulaatio toimi reaaliajassa WMS -backendin simulaatiomoduulilta saadun datan perusteella. Myyntivisualisaatio toteutettiin yksinkertaisena animaationa järjestelmän toiminnasta erillisillä ohjelmakoodeilla, jossa hissi suoritti tehtäviä ennalta määritetyissä lokaatioissa.



Kuva 9. 3D-layout upotettuna WMS-selainversioon.



Kuva 10. 3D-layout sivusta päin kuvattuna (ortografinen kamera).

Toimintamallien vertailu osoitti, että nykyinen toimintamalli jakautuu sopimusta edeltävään ja sen jälkeiseen vaiheeseen. Sopimuksen jälkeisessä vaiheessa myyntivaiheen simulaatioita, visualisaatioita ja malleja ei enää hyödynnetä, vaan WMS-kehitys aloitetaan kokonaan omana projektinaan. Uuden toimintatavan tuoma hyöty olisi sen suoraviivaisuus, jossa lähtötilanteena on yksi 3D-malli, josta kehitetään Unity-ympäristössä myynti- ja WMS-simulaatiot. Taulukossa 1 on koottuna yhteenveto toimintamallien vertailusta.

Taulukko 1. Toimintamallien vertailu.

Vertailukohde	**Nykyinen toimintamalli**	**Uusi toimintamalli**
Lähtötilanne	3D-malli tai Autocad-pohjapiirustus (2D), joissain tapauksissa molemmat	3D-malli
Myyntivaiheen simulointi	Autocad-piirustuksen pohjalta Visual Componentsilla	3D-mallin pohjalta Unityllä
WMS-simulointi	Autocad-piirustuksen pohjalta SVG	3D-mallin pohjalta Unityllä
Hyödyt	Vakiintunut käytäntö, osaamista jo ennalta	Suoraviivaisempi toteutustapa nykyiseen verrattuna
Haitat	Monivaiheinen, työläs ja kallis prosessi	Käyttöönotto vaatii muutoksia yrityksen sisäisiin toimintatapoihin

Tutkimusten perusteella voidaan todeta, että uusi toimintamalli on toteutettavissa. Tulokset osoittavat, että toimintatapaa muuttamalla, simulaatio- ja käyttöliittymäkehitysprosessia on mahdollista tehostaa ja sen työvaiheita karsia. Jotta tavoitteeseen päästäisiin, tulisi yrityksen sisäisiin toimintatapoihin kuitenkin tehdä muutoksia, sillä nykyisen prosessin toteutustapa on tarpeettoman monivaiheinen, työläs ja hidas. Muutosten tekeminen edellyttää aluksi nykyisten toimintatapojen perusteellisen kartoittamisen. Sen jälkeen on mahdollista hahmottaa, millaisia toimenpiteitä uuteen toimintatapaan siirtyminen vaatisi. Tulosten perusteella voidaan kuitenkin jo sanoa, että 3D-mallinnus tulisi tehdä varhain projektin myyntivaiheessa, jolloin malli olisi käytettävissä Unity-projektia varten. Lisäksi eri vaiheiden työntekijöiden tulisi yhteisesti sopia toteutustavoista, jolloin prosessista saataisiin mahdollisimman sujuva.

8 YHTEENVETO

Tämän työn tavoitteena oli kartoittaa mahdollisuuksia tehostaa Pesmel Oy:n simulointi- ja käyttöliittymäkehitysprosessia sekä suorittaa vertailu nykyisen ja uuden toimintatavan välillä. Tarkoituksena oli selvittää, voisiko Unity-pelimoottorin käyttö uutena työkaluna olla hyödyllistä prosessin toteutuksessa. Lisäksi kartoitettiin yrityksen nykyisiä toimintatapoja toteutuksen osalta.

Työn alussa selvitettiin nykyisen toteutuksen vaiheet, jotka tarkentuivat työn edetessä. Uudessa toteutuksessa lähdettiin liikkeelle tutustumalla Unityyn ja sen ominaisuuksiin. Tämän jälkeen alkoi varsinainen kehitysvaihe, jossa ensimmäinen askel oli tuoda työtä varten optimoitu 3D-malli Unity-ympäristöön ja tehdä sille tarvittavat toimenpiteet.

Ensimmäiseksi toteutettiin Unity-mallin kytkeminen WMS-simulaattoriin, jossa laitteiden sijainti- ja tehtävätieto haettiin WebSocketin ja rullatieto REST-rajapinnan avulla WMS-backendilta. Laitteet ohjelmoitiin liikkumaan backendilta saadun datan perusteella. Mallia optimoitiin ja kehitettiin edelleen työn edetessä, mutta opinnäytetyön kannalta olennaiset ominaisuudet saatiin kehitettyä jo melko varhaisessa vaiheessa toteutusta. Myyntivisualisaatio tehtiin erillään WMS-simulaatiosta Git-versionhallintajärjestelmän omassa haarassa, jolloin sitä oli mahdollista kehittää ilman, että se vaikutti pääprojektiin. Siinä laitteiden liikkeet animoitiin vastaamaan todellisen järjestelmän toimintaa erillisiä ohjelmakoodeja käyttäen.

Työhön liittyvät haasteet koskivat pääasiassa uuden ohjelmiston ja ohjelmointikielen opettelua. Toteutuksen eri vaiheet vaativat kokeiluja, joiden avulla saatiin selville kannattavat toimintatavat. Työssä päästiin tavoitteisiin, eli saatiin luotua vertailu nykyisestä ja uudesta toimintamallista sekä toteutettua myyntivisualisaatio ja WMS-simulaatio Unity-pelimoottorilla. Lähes kaikki halutut ominaisuudet saatiin kehitettyä Unity-projektiin työn toteutuksen puitteissa. Jos uusi toimintamalli otetaan tulosten perusteella käyttöön, tulee sitä kuitenkin jatkokehittää ja optimoida edelleen, jotta sen toteutuksesta saadaan mahdollisimman sujuva.

LÄHTEET

Bandyopadhyay, S., & Bhattacharya, R. (2014). *Discrete and continuous simulation: theory and practice.* Taylor & Francis. https://doi.org/10.1201/b17127

Bangsow, S. (2020). *Tecnomatix Plant Simulation: Modeling and programming by means of examples* (2. painos). Springer International Publishing. https://doi.org/10.1007/978-3-030-41544-0

Banks, J. (2005). *Discrete-event system simulation* (4. painos). Pearson Prentice Hall.

Bromberg, M. (17.10.2024). Unity 6 Launches today! *Unity Blog.* https://unity.com/blog/introducing-unity-6-launch

Diaconu, A. (17.5.2024). *The WebSocket API and protocol explained.* Ably. Haettu 2.12.2024, https://ably.com/topic/websockets#web-sockets-the-web-socket-protocol-and-api-explained

Ekins, B. (2008). *How deep is the rabbit hole? Examining the matrix and other Inventor® Math and Geometry Objects.* Autodesk University. https://modthemachine.typepad.com/files/mathgeometry.pdf

Hatton, P. (16.10.2023). *Unity: everything you need to know.* Creative Bloq. Haettu 21.10.2024, https://www.creativebloq.com/features/unity-everything-you-need-to-know#section-unity-what-is-it

IBM. (5.8.2021). *What is a digital twin.* Haettu 18.11.2024, https://www.ibm.com/topics/what-is-a-digital-twin

Inkscape. (i.a.). *About: Inkscape overview.* Haettu 18.11.2024, https://inkscape.org/about/

Laurila, H. (i.a.). *Teollisuusympäristön visualisointi Visual Components -toteutuksena* [valokuva].

Levy, S. (20.12.2024). *Graphical user interface.* Britannica. Haettu 4.1.2025, https://www.britannica.com/technology/graphical-user-interface

Mátrai, R. (2010). *User interfaces.* IntechOpen. https://doi.org/10.5772/230

Maunula, J. (Product Owner ICT, Pesmel). (18.11.2024). *WMS-esittely* [asiantuntijahaastattelu].

Mäki-Jussila, T. (2024a). *Yksinkertaistettu 3D-malli varastohyllyn päätyrakenteesta* [valokuva].

Mäki-Jussila, T. (2024b). *Hyllystöhissin alkuperäinen (vas.) ja muokattu 3D-malli* [valokuva].

Mäki-Jussila, T. (2024c). *Esimerkki pääkokoonpanosta* [valokuva].

Korhan, O. (2023). *Digital twin technology*. IntechOpen. https://doi.org/10.5772/intechopen.113345

Pesmel. (2024). *About us.* Haettu 14.10.2024, https://pesmel.com/about-us/

Pesmel. (2025). *Warehouse Management System - Pesmel WMS specification.* M-Files.

Red Hat. (8.5.2020). *What is a REST API?* Haettu 9.12.2024, https://www.redhat.com/en/topics/api/what-is-a-rest-api

Richards, G. (2022). *Warehouse management: The definitive guide to improving efficiency and minimizing costs in modern warehouse* (4. painos). Kogan Page.

Ritter, M., & Winterbottom, C. (2017). *UX for the web: Build websites for user experience and usability*. Packt.

Rubin, J., & Chisnell, D. (2008). *Handbook of usability testing: How to plan, design, and conduct effective tests* (2. painos). Wiley Pub.

Rosencrance, L. (16.8.2024). *Graphical User Interface (GUI).* Techopedia. Haettu 29.11.2024, https://www.techopedia.com/definition/5435/graphical-user-interface-gui

Rossetti, M. D. (2016). *Simulation modeling and arena* (2. painos). Wiley.

SAP. (i.a.) *What is a warehouse management system (WMS).* Haettu 2.12.2024, https://www.sap.com/products/scm/extended-warehouse-management/what-is-a-wms.html

Suomen asiakastieto. (i.a.). *Pesmel Oy*. Haettu 16.10.2024, https://www.asiakastieto.fi/yritykset/fi/pesmel-oy/21203130/yleiskuva

Uitop. (10.10.2024). *How to design a warehouse management system (WMS): Essential guide.* Haettu 29.1.2025, https://uitop.design/blog/product/wms-design/

Unity Technologies. (i.a.-a). *Unity Learn: Introduction to Digital Twins with Unity.* Haettu 18.11.2024, https://learn.unity.com/tutorial/introduction-to-digital-twins-with-unity?uv=2022.3

Unity Technologies. (i.a.-b). *Solutions for industry: Manufacturing.* Haettu 17.10.2024, https://unity.com/solutions/manufacturing

Unity Technologies. (i.a.-c). *Unity Manual: Importing a model.* Haettu 17.10.2024, https://docs.unity3d.com/Manual/ImportingModelFiles.html

Unity Technologies. (i.a.-d). *Unity Manual: Rotation and orientation in Unity.* Haettu 18.10.2024, https://docs.unity3d.com/Manual/QuaternionAndEulerRotationsInUnity.html

Unity Technologies. (i.a.-e). *Unity Manual: Preparing your model files for export.* Haettu 18.10.2024, https://docs.unity3d.com/Manual/models-preparing.html

Unity Technologies. (31.7.2018). *Unity Manual: Unpacking Prefab instances.* Haettu 9.11.2024, https://docs.unity3d.com/Manual/UnpackingPrefabInstances.html

Unity Technologies. (5.6.2020). *Unity Manual: Transforms.* Haettu 9.11.2024, https://docs.unity3d.com/2017.4/Documentation/Manual/Transforms.html

Visual Components. (i.a.). *Visual Components: About us.* Haettu 21.10.2024, https://www.visualcomponents.com/about-us/

Visual Components. (28.2.2017a). *Visual Components Academy: Component scripting*. Haettu 28.10.2024, https://academy.visualcomponents.com/lessons/component-scripting/

Visual Components. (26.1.2017b). *Report Statistics* [video]. YouTube. Haettu 28.10.2024, https://www.youtube.com/watch?v=7sMyptzTXdA&t=185s

Visual Components. (1.3.2017c). *Visual Components Academy: Component states.* Haettu 28.10.2024, https://academy.visualcomponents.com/lessons/component-states/

Visual Components. (2024). *Visual Components Experience guide.* https://visualcomponents.com/wordpress/wp-content/uploads/2024/08/vc-experience-guide-1.7.pdf

Werner, S. (25.11.2024). MES vs. ERP: Do I need both in my factory? Pico. Haettu 31.1.2025, https://www.picomes.com/resources/blog/mes-vs-erp-the-differences-and-benefits

Zenva. (19.9.2024). *What is Unity? – a top game engine for video games.* Haettu 21.10.2024, https://gamedevacademy.org/what-is-unity/#What_is_Unity_and_Who_Owns_Unity_Game_Engine

Ämmälä, J. (ICT-suunnittelija, Pesmel). (1.10.2024). *WMS-layoutkehittämisprosessi* [asiantuntijahaastattelu].