Saku Rantalainen

Raskaskonesimulaattorin immersion parantaminen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

18.4.2023

Tiivistelmä

Tekijä:	Saku Rantalainen
Otsikko:	Raskaskonesimulaattorin immersion parantaminen
Sivumäärä:	32 sivua
Aika:	18.4.2023
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Suuntautumisvaihtoehto:	Pelisovellukset
Ohjaaja:	Lehtori Miikka Mäki-Uuro

Insinöörityön tavoitteena oli parantaa jo olemassa olevan kaivinkonesimulaattorin äänimaailmaa ja käyttöliittymää immersion parantamiseksi. Kaivinkoneen käyttöliittymän parantaminen aloitettiin syksyllä 2022 Metropolia Ammattikorkeakoulun innovaatioprojektilla. Opinnäytetyön tarkoitus oli viedä innovaatioprojektin aloittama kehitystyö loppuun ja tuoda simulaattoriin täysin uudistettu äänimaailma. Työ tehtiin kone- ja prosessisuunnittelualan yritykselle.

Projekti tehtiin kahdessa osassa, ensin käyttöliittymä, sitten äänet. Todellisuudessa molempia projekteja tehtiin päällekkäin, vuoroviikoin. Aikataulullisesti äänimaailman suunnitteluun ja tekemiseen varattiin enemmän aikaa, sillä käyttöliittymän suunnittelu ja kehitystyö oli jo aloitettu. Molemmille osille oli varattu 7–8 viikkoa aikaa, mukaan lukien suunnittelu- ja tutkimustyö.

Projektissa käytettiin yrityksen omien sovellusten lisäksi Unity-pelimoottoria ja Reaper-audiotyöasemaa. Projektin toteutettiin etätyönä, muutamaa vierailua yrityksen tiloissa lukuun ottamatta.

Projektin tavoitteet saavutettiin, mutta käyttöliittymää ei saatu vielä testattua oikealla simulaattorilla. Tuloksena oli kuitenkin toimiva käyttöliittymä, jota jatkokehitetään edelleen ja uusi äänimaailma, joka sai erittäin hyvän vastaanoton.

Projektiin tehdään tulevaisuudessa päivityksiä. Uudet mittarit ja napit ovat varmasti ensimmäisten päivitysten joukossa. Myös parempi pohja on tarkoitus lisätä. Kamera-näkymä oli ollut kehitteillä jo ennen tämän projektin alkua, mutta siihen liittyvä teknologia on vielä kehitteillä.

Avainsanat: simulaattori, immersio, Unity

Abstract

Author:	Saku Rantalainen
Title:	Improving heavy-duty simulator immersion
Number of Pages:	32 pages
Date:	18 April 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Specialisation option:	Game Applications
Instructor:	Miikka Mäki-Uuro, Senior Lecturer

The project aimed to improve the soundscape and user interface of the existing excavator simulator to improve immersion. The improvement of the excavator's user interface was started in autumn 2022 with an innovation project. The purpose of this project is to complete the development work started with the innovation project and bring a completely renewed soundscape to the simulator. The project is written in two parts, first the interface, then the sounds. Both projects were done at the same time.

In addition to the company's own applications, the project used the Unity game engine and the Reaper audio workstation. The project was implemented remotely, except for a few visits to the company's premises.

The goals of the project were achieved, but the user interface was not tested in a real simulator. The result was a functional base for the interface that will be further developed and a new soundscape that was very well received. The future of the project is quite clear and there will be future updates. New gauges and buttons will be among the first updates. Better visuals are also planned to be implemented.

Keywords: simulator, immersion, Unity

Sisällys

1 Johdanto 1

2 Projektin lähtökohdat 2

2.1 Käyttöliittymä innovaatioprojektina 4
2.2 Kaivinkonesimulaattorin äänimaailma 5

3 Käytettävien teknologioiden valinta 6

3.1 Mevea Simulation -sovellus 6
3.2 Unity-pelimoottori 8
3.3 Äänimoottorin valinta 9

4 Käyttöliittymän ja äänimaailman toteutus 12

4.1 Käyttöliittymä 12
4.2 Äänimaailma 19
4.2.1 Kaivinkoneen äänet 19
4.2.2 Taustaäänet 26

5 Projektin lopputulos 29

5.1 Käyttöliittymä 29
5.2 Äänet 31
5.3 Tulevaisuus 31

6 Yhteenveto 32

Lähteet 33

1 Johdanto

Raskaskonesimulaattoriksi voidaan luokitella lähes kaikki isojen työkoneiden simulaattorit. Nämä simulaattorit ovat hyviä työkaluja tuotekehitykseen, testaamiseen, koulutukseen ja tuotetukeen isoille konevalmistajille. Hyvin suunniteltu ja toteutettu simulaatioratkaisu palvelee koneen koko elinkaaren ajan.

Insinöörityön tavoitteena oli parantaa jo olemassa olevan kaivinkonesimulaattorin äänimaailmaa ja käyttöliittymää immersion parantamiseksi. Immersio on metaforinen ilmaus virtuaalitodellisuuteen ”uppoamisesta”. Sillä viitataan käyttäjän kokemukseen syventyä tässä tapauksessa simulaattoriin niin, että käyttäjän keskittyminen on simulaattorissa eikä hän välttämättä tiedosta simulaation hetkellä ulkopuolista maailmaa. (1.) Kaivinkoneen käyttöliittymän parantaminen aloitettiin syksyllä 2022 Metropolian innovaatioprojektilla. Projektin tavoitteet saavutettiin, mutta käyttöliittymää ei saatu vielä testattavaksi asti. Insinöörityönprojektin tarkoitus on viedä innovaatioprojektin aloittama kehitystyö loppuun ja tuoda simulaattoriin täysin uudistettu äänimaailma.

Insinöörityö toteutettiin Mevea Oy:n sisäisenä projektina. Mevea (Mechatronic and Virtual Engineering Applications) on vuonna 2005 perustettu yritys, jonka perusti Lappeenrannan teknillisen yliopiston mekatroniikan ja reaaliaikaisen simuloinnin parissa työskentelevä seitsemän hengen tutkijaryhmä. Jo kahden vuoden reaaliaikaisen simulaatio-ohjelmiston kehittämisen jälkeen Mevea aloitti nyt jo monta vuotta kestäneen pitkäaikaisen yhteistyön kaivostyökoneita valmistavan Sandvikin kanssa, joka oli samalla Mevean ensimmäinen globaali asiakas. Ensimmäinen projekti sisälsi muun muassa Sandvikin pyöräkuormaimen ja kaivosporauslaitteen simulaatiomallit. (2.)

Yritys oli kasvanut vuoteen 2008 mennessä kansainvälisesti jo niin suureksi, että ensimmäinen iso satamanosturin ohjaamosimulaattori toimitettiin lastinkäsittelyratkaisuja ja -palveluja satamiin, terminaaleihin, jakelukeskuksiin ja ras-

kaan teollisuuden käyttöön valmistavalle Kalmarille koulutustarkoituksiin Brasiliaan. Vielä tällöin simulaattoreiden kaikki kokoonpanotyöt tehtiin Mevean sisällä. Nämä kaivoskoneiden simulaattorit ovat erittäin tärkeä osa Mevean liiketoimintaa. Mevea toimitti maanalaisten rakennus- ja kaivosprosessien toimialalla toimivalle Normetille betoninruiskutus- ja skaalauskoneiden simulaattoreita vuonna 2012. Nämä simulaattorit sisälsivät reaaliaikaisen hydraulisen simulaation ja konfiguroitavan ohjausjärjestelmän integroinnin, jotka ovat Mevean simulaattoreiden ainutlaatuisia ominaisuuksia. (2.)

Vuonna 2018 Mevean isoimpia saavutuksia oli, kun se valittiin Siemensin yhteistyökumppaniksi. Päätökseen vaikuttivat muun muassa Mevean reaaliaikaisen simulointiohjelmiston tarkkuus, että sen kyky integroitua saumattomasti Siemensin SIMOCRANE-teknologia-alustaan. Tämä alusta tuotti digitaalisia kaksosia, jotka pystyvät simuloimaan esimerkiksi oikeiden nostureiden käyttäytymistä ja käyttöä.

Nykyään Mevea auttaa asiakkaita ympäri maailmaa innovoimaan mahdollistaen digitaalisten kaksosten teknologian tuotteen elinkaaren ajan eri toimialoilla. (2.)

Luvussa 2 esitellään molempien projektien lähtökohdat ja se, miten niitä lähdettiin toteuttamaan. Luku 3 keskittyy käytettyihin teknologioihin ja niiden valintaan, kuten esimerkiksi äänimoottorin valinta. Luvussa 4 kuvataan molempien osien toteutus alusta loppuun, alkaen käyttöliittymästä ja lopuksi äänien toteutus. Luvussa 5 projektista poimitaan onnistumiset ja parantamiset.

2 Projektin lähtökohdat

Insinöörityö jaettiin kahteen osaan: Ensimmäisessä osassa keskityttiin innovaatioprojektissa aloitetun HMI-pohjan (kuva 1) tekemiseen. Human-Machine Interface eli HMI tarkoittaa käyttöliittymää tai ohjausnäkymää, jolla ”yhdistetään” käyttäjä koneeseen, järjestelmään tai laitteeseen. Vaikka termiä voidaan teknisesti soveltaa mihin tahansa näyttöön, jonka avulla käyttäjä voi olla vuorovaiku-

tuksessa laitteen kanssa, käyttöliittymää käytetään yleisimmin teollisen prosessin yhteydessä (3). Työn toisessa osassa tehtiin kaivinkonesimulaattorin uusi äänimaailma. Molemmat osa-alueet vaikuttavat suuresti simulaattorin yleiseen immersion tunteeseen.

Kuva 1. Mevean vanha HMI-pohja, jossa näkyvillä eri hallintanappeja.

HMI-pohjaa lähdettiin työstämään Metropolia Ammattikorkeakoulun innovaatioprojektin lopussa pidetyn palaverin pohjalta. Innovaatioprojektissa olivat mukana Mevealta asiakasprojektien päällikkö Vesa Ahola ja tekninen avustaja Toni Uronen. Molemmat olivat mukana myös tässä projektissa. Vesa Ahola toimi myös tämän projektin vetäjänä. Toni Uronen toimii Mevealla ohjelmistokehittäjänä.

Projektin aikataulu oli alusta asti tiukka. Projektin piti alkaa jo vuoden 2022 joulukuussa, mutta sitä jouduttiin siirtämään Mevean muiden projektien ja tilausten takia. Projekti saatiin aloitettua tammikuussa 2023, mutta samat rajoitteet varjostivat tätäkin ajankohtaa. Tästä syystä projektin laajuus haluttiin pitää suhteellisen pienenä. Koodista ja uusista ominaisuuksista haluttiin tehdä yksinkertaisia ja helposti implementoitavia mahdollisesti myös muihin simulaattoreihin.

2.1 Käyttöliittymä innovaatioprojektina

Ennen insinöörityötä tehdyssä innovaatioprojektissa keskityttiin uuden nykyaikaisemman käyttöliittymän (kuva 2) tekemiseen, joka tuli käyttöön Mevean omaan kaivinkonesimulaattoriin. Jo alussa päätettiin uuden käyttöliittymän uudesta ominaisuudesta, joka mahdollistaa mittareiden ja muiden hallintalaitteiden paikan vaihdon ruudulla. Näin yhtä pohjaa voidaan käyttää useissa eri simulaattoreissa. Projektiin kuului myös laaja tutkimustyö, jossa tutkittiin eri konevalmistajien ratkaisuja ja ominaisuuksia pääosin kaivinkoneiden ja pyöräkuormaajien osalta. Tutkimustyön pohjalta laadittiin lista tarvittavista komponenteista eli mittareista, napeista jne.

Kuva 2. Innovaatioprojektin aikana tehty geneerinen HMI-malli. Mallissa oli paikka kahdelle mittarille ja yhdelle vapaavalinnaiselle ruudulle.

Innovaatioprojektin alkuperäisiin tavoitteisiin kuului

- selvittää eri valmistajien kaivinkoneissa, pyöräkuormaajissa ja lukkinostureissa käytettyjen mittareiden, indikaattoreiden ja ohjauksien erilaisia toteutuksia
- määritellä ja dokumentoida tutkimuksen tulokset visuaalisesti ja toiminnallisesti
- laatia näistä toteutusehdotus, joka palvelee Mevean geneerisiä työkoneita
- toteuttaa valittujen komponenttien ja näyttöjen visuaalinen ja toiminnallinen toteutus.

Viimeisteltyyn komponenttilistaan kuuluivat muun muassa

- eri jäähdytin- ja hydrailunesteiden lämpötilan mittarit
- nopeus- ja kierrosmittarit
- ekologisuutta tarkkaileva mittari (mittarin teknisestä toteutuksesta ei ollut vielä varmuutta, mutta pohja haluttiin jo valmiiksi)
- mahdollinen valikkojärjestelmä, jossa voidaan vaihtaa esim. mahdollisen peruutuskameran ja inforuudun välillä.

Innovaatioprojektin aikana HMI-pohjaa ei testattu Mevean omilla simulaattoreilla. Testaus suunniteltiin toteutettavaksi insinöörityön aikana. Mahdollista valikkojärjestelmä ei tehty, koska sitä ei katsottu tarpeelliseksi eikä sitä olisi saatu tehtyä ajoissa. Muut tavoitteet saatiin suoritettua ajallaan.

2.2 Kaivinkonesimulaattorin äänimaailma

Insinöörityöprojektin toisessa osassa keskityttiin kaivinkonesimulaattorin äänimaailman parantamiseen. Tähän liittyi taustaäänien lisääminen ja vanhojen kaivinkoneen äänien parantaminen. Uudet äänet arkistoidaan Mevean omaan järjestelmään, jotta niitä voidaan käyttää mahdollisesti myös muissa simulaattoreissa.

Projektin alkuvaiheessa päätettiin tehdyn tutkimustyön pohjalta äänimoottorin valinnasta ja mahdollisista äänikirjastoista. Käytiin myös läpi simulaattorin jo olemassa olevia äänitiedostoja ja niitä ohjaavia koodipätkiä. Simulaattorissa ei aloitushetkellä ollut mitään muita ääniä kuin kippirekan ja kaivinkoneen äänet. Simulaattoriin piti siis suunnitella ja toteuttaa täysin alusta alkaen uudet taustaäänet.

Mevean kaivinkonesimulaattori (kuva 3) toimii satamaympäristössä, mikä piti ottaa huomioon varsinkin taustaäänien suunnitteluvaiheessa. Uusien äänien äänityksestä satamaympäristössä keskusteltiin. Yhdeksi äänityspaikaksi ehdotettiin Vuosaaren satamaa, joka on merkittävä kontti- ja perävaunuliikenteen satama. Vuosaaren satama olisi tästä syystä soveltunut hyvin työmaa-aiheiseen ympäristöön. Äänityksistä kuitenkin luovuttiin pääosin tarpeeksi hyvien äänikirjastojen takia, mutta myös aikarajoitteet olivat iso syy äänityksistä luopumiseen.

Kuva 3. Mevean kaivinkonesimulaattori, joka pohjautuu Hitachin ZX240G-kaivinkoneen malliin, johon uudet äänet lopuksi lisättiin.

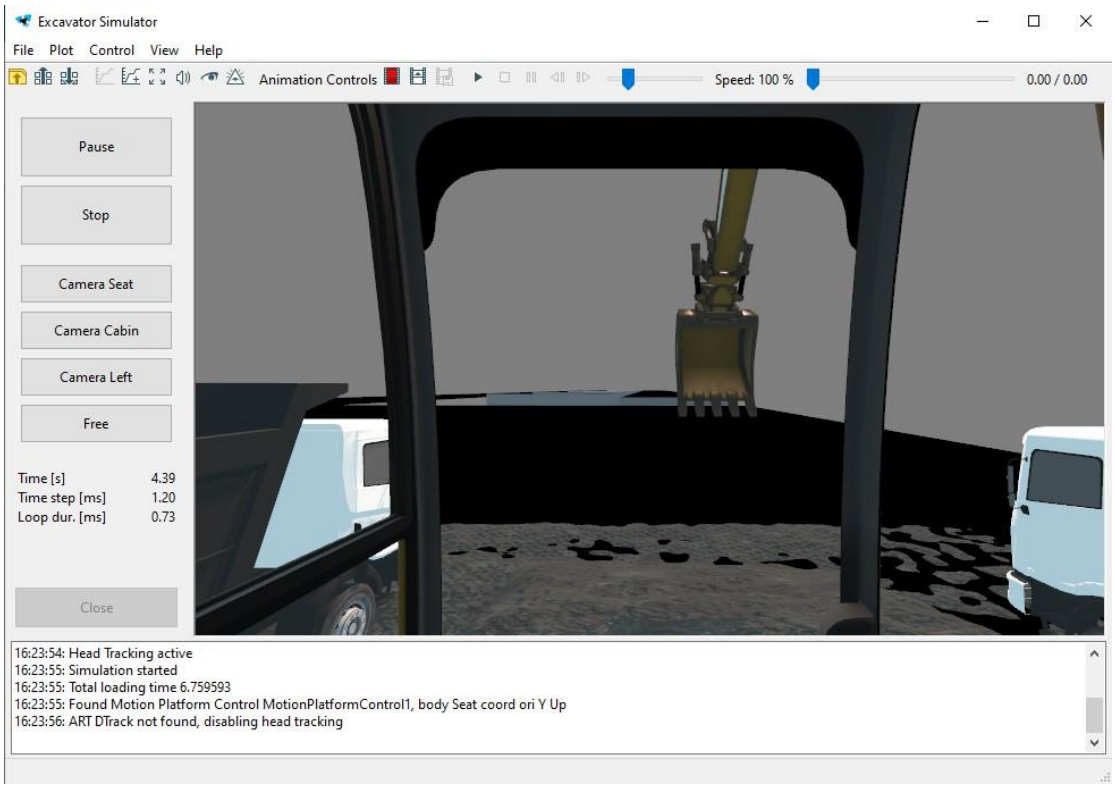
3 Käytettävien teknologioiden valinta

Tässä luvussa kuvataan projektissa käytetyt teknologiat. Alaluvuissa kuvataan esimerkiksi äänimoottorin valintaa ja Mevean omia työkaluja.

3.1 Mevea Simulation -sovellus

Mevea Simulation Software (kuva 4) on Mevean itsekehittämä reaaliaikaisen simuloinnin työkalu, joka on kehitetty ohjelmistokehittäjille. Sen avulla ohjelmistokehittäjät voivat testata työkoneen uusia ominaisuuksia jo olemassa oleviin malleihin. Ohjelmiston ydin on Mevean oma fysiikkamoottori, joka simuloi tarkasti koneen mekaniikkaa, hydrauliikkaa, voimansiirtoa ja käyttöympäristöä. (4.)

Mevea Simulation Softwaren yhtenä osana on Mevea Solver -sovellus, jolla voidaan käyttää eri työkoneidenmalleja reaaliajassa Mevean simulaatiomoottorilla. Muita sovelluksen ominaisuuksia ovat esimerkiksi käyttäjän valitsemien parametrien tuloksien tarkastelu ja analysointi ja tallennusmahdollisuus koko simulaatio tapahtumasta myöhempää tarkastelua ja tutkimusta varten. (4.)



Kuva 4. Mevea Solver, joka on osa Mevea Simulation Softwarea, jossa käynnissä kaivinkoneen simulaatio.

Mevea Simulation Softwaren avulla voidaan rakentaa fysiikkaan perustuvia digitaalisia kaksosia, jotka pystyvät simuloimaan eri valmistajien mallien käyttäytymistä ja käyttöä reaaliajassa, mikä auttaa saavuttamaan asiakkaan yritykselle merkittäviä liiketoimintahyötyjä tuotteen koko elinkaaren ajan. Monet maailmanluokan yritykset ovat todenneet sen toimivan käytännössä. Dynamiikan taustalla olevat matemaattiset laskelmat on tieteellisesti todistettu. (4.)

Mallinnusvaiheessa simulaatiomalli rakennetaan samalla tavalla kuin useissa CAD-ohjelmistoissa. Massa- ja inertiatiedot tuodaan CAD-mallista koneen runkojen luomista varten. Nämä kappaleet kiinnitetään sitten toisiinsa erilaisilla rajoituksilla ja liitoksilla. Hydrauliikka lisätään tässä vaiheessa, samoin kuin moottori ja voimansiirto. Runkoja voidaan simuloida myös joustavasti, ja kitkakäyrien mallintaminen kohteiden välillä on helppoa. (4.)

Kun parametrit on asetettu, reaaliaikainen simulointi voidaan suorittaa välittömästi. Mevea Simulation Software sisältää tarkat reaaliaikaiset tiedot sekä simulaation aikana että myös myöhempää tutkimusta varten. Käyttäjä voi piirtää tietoja esimerkiksi seuraavista asioista:

- rungon asento ja suunta
- energiankulutus
- rajoitusvoimat
- hydrauliset paineet
- runkojen väliset törmäykset.

(4.)

Mevea Software API ja muut rajapinnat mahdollistavat useita mahdollisuuksia muodostaa yhteys olemassa olevaan kehitysympäristöön. Tässä projektissa Solver yhdistyy Unityssä rakennettuun simulaatioympäristöön. Simulaation ollessa käynnissä Solver lähettää Unitylle tietoja esim. koneen liikkeistä, sijainnista ja kaivutapahtumasta ympäristössä. Solver toimii tässä tapauksessa siis yhdessä Unityn kanssa lähettäen tietoja Unitylle ja vastaanottaa niitä siltä. Esimerkiksi kaivinkoneen mittarit toimivat siten, että Solver kirjoittaa ja lähettää tietoa siten, että mittari, joka on erillisessä Unity-projektissa, pystyy näyttämään oikeat arvot. Solveria käytetään tässä projektissa vain äänien testaukseen ja simulaattorin hallintaan.

3.2 Unity-pelimoottori

Vuonna 2005 julkaistu Unity on Unity Technologiesin kehittämä monialustainen pelimoottori, joka julkistettiin ensimmäisen kerran Apple WorldWide Developers Conferencessa Mac OS X -pelimoottorina. Moottoria on julkaisun jälkeen laajennettu asteittain tukemaan erilaisia konsoli-, työpöytä-, mobiili- ja virtuaalitodellisuusalustoja. Se on saanut jalansijaa erityisesti suosittujen iOS- ja Android-mobiilialustojen pelien kehittämisessä. Unityä pidetään helppokäyttöisenä aloitteleville kehittäjille, minkä takia se on erittäin suosittu esimerkiksi indiepelien kehittämisessä. (5.)

Mevea Solverin ohella Mevean simulaattorit käyttävät Unityä simulaattorin ympäristön mallinnukseen ja ohjaukseen. Sekä HMI-pohja että kaivinkonesimulaattori ovat omia Unity-projekteja. Versionhallintaa ei käytetty tässä projektissa, eli Mevealta lähetettiin vanhat projektit ja projektin lopuksi valmiit projektit lähetettiin takaisin, ja ne korvattiin Mevean järjestelmässä.

Unity käyttää ohjelmointikielenään pääasiassa C#-koodikieltä, joten sitä käytettiin myös tässä projektissa. Microsoftin kehittämä C# on moderni, oliosuuntautunut ja tyyppiturvallinen ohjelmointikieli, jonka avulla ohjelmistokehittäjät voivat rakentaa monia erityyppisiä, turvallisia ja kestäviä sovelluksia, jotka toimivat Microsoftin .NET-alustalla. C#:n kuuluu C-kieliperheeseen, mikä auttaa ohjelmoijia, joille jo tuttuja ohjelmointikieliä ovat muut C-perheeseen kuuluvat kielet C sekä C++, sekä myös Java- ja JavaScript-ohjelmoijia. C# julkaistiin kesäkuussa vuonna 2000. Sen haluttiin yhdistävän C++:n tehokkuus ja Visual Basicin tuottavuus. (6.)

3.3 Äänimoottorin valinta

Äänimoottorin valintaan vaikuttivat suuressa määrin sen helppokäyttöisyys ja nopea käyttöönotto ottaen huomioon projektin aikataulu sekä mahdolliset kustannukset. FMOD, Wwise ja Unityn oma äänimoottori olivat suunnitteluvaiheessa esillä, joista kaksi ensimmäistä ovat lisensoitavia tuotteita. Wwise jätettiin pois heti alussa, pääosin huonon dokumentaation ja insinöörityön tekijän vähäisen tietämyksen takia.

Molemmat jäljelle jääneet äänimoottorit tarjoavat samanlaiset perustyökalut. Sekä Unity että FMOD mahdollistavat ääniraitojen tuomisen ja niille sellaisten sääntöjen luomisen, joiden mukaan soittaa tiettyjä ääniraitoja, lisätä eri äänitehosteita, miksata ja muodostaa tietynlainen hierarkia sekä seurata äänijärjestelmän suorituskykyä. Kuitenkin teknisten ominaisuuksien määrässä ja järjestelmän joustavuudessa FMOD tarjoaa enemmän. Unitystä puuttuvat vaihtoehdot

voidaan joissain määrin korvata hyvällä suunnittelulla. Kaiken kaikkiaan molemmissa moottoreissa on tarvittavat ominaisuudet toimivan äänijärjestelmän luomiseksi simulaattoriin.

Yksi FMOD:n suurimmista eduista ja syy siihen, miksi se haluttiin alun perin myös tähän projektiin, on sen tapahtumaeditori. Sen aikajananäkymä antaa vapauden muokata ja käsitellä äänisisältöä myös sen jälkeen, kun se on tuotu äänimoottorille soitettavaksi. Myös yksittäisiä äänitehosteita voi yhdistää useammalla tasolla. Esimerkiksi sen sijaan, että käyttäisi muunnelmia kaivinkoneen moottorin äänistä ja niitä soitettaisiin satunnaisesti, on mahdollista tuoda muunnelmia moottorin äänistä: käynnistys, kierrosten nousu, ”tasainen” moottorin ääni, kierrosten lasku ja lopuksi sammutus. Nämä kerrokset voidaan asettaa yhdistettynä itse audiomoottoriin, mikä luo enemmän joustavuutta ja mahdollisuuksia hienosäätöön. Jokaiseen eri kerrokseen on mahdollista liitää erillisiä äänitehosteita, jotka ovat erillään muusta kokonaisuudesta. Tämä koko prosessi on mahdollista suorittaa FMOD:n sisäänrakennetuilla työkaluilla ilman ohjelmointia, kun taas Unityssa jopa alkeellisinta ja yksinkertaisten äänileikkeiden satunnaistaminen on tehtävä koodin avulla. FMOD:n aikajana mahdollistaa myös musiikin tai tämän projektin tapauksessa taustaäänien ja äänitehosteiden sekvensoinnin jaksosta toiseen, mikä ei ole helppoa toteuttaa Unityssa käyttämättä useita eri äänilähteitä ja ohjelmointia niiden laukaisemiseen ja pysäyttämiseen oikeaan aikaan. (7.) Sekvenssillä tarkoitetaan jonkin musiikillisen aiheen tai teeman toistamista eri sävelkorkeuksilta (8).

Vaikka FMOD:n dokumentaatio on selkeä ja parempi kuin esimerkiksi Wwisen, olisi sen liittäminen tähän projektiin vienyt liikaa aikaa. FMOD ei ollut tuttu kaikille simulaattoreiden parissa työskenteleville yrityksen sisällä, mikä olisi voinut tuottaa ongelmia jatkossa. Tämä ei kuitenkaan sulje pois mahdollisuutta vaihtaa jatkossa FMOD:iin, mikäli se nähdään taloudellisesti hyväksi sijoitukseksi.

Suurin syy äänimoottorin valintaan olivat kuitenkin FMOD:n lisenssimaksut. FMOD:n lisenssin hinta määräytyy yrityksen liikevaihdon mukaan. FMOD:lla on

kolme eri hintaluokkaa: Indie, Basic ja Premium. Indielisenssin maksimi liikevaihto on 500 000 dollaria, jolloin lisenssin hinta on ilmainen tai enintään 2 000 euroa. Basic-lisenssin liikevaihto on vähintään 500 000 dollaria ja enintään 1 500 000 euroa, jolloin lisenssin hinta on 5 000 euroa. Premium-lisenssin saamiseksi liikevaihdon täytyy olla yli 1 500 000 dollaria, jolloin lisenssin hinta on 15 000 euroa. Edellä esitetty hinnoittelu ei kuitenkaan koske sijaintiin perustuvaa viihdettä (Location-based entertainment), simulaattoreita, sulautettuja järjestelmiä, asennuksia ja pelijärjestelmiä (mm. kasinot). (9.) Mevean tapauksessa lisenssi olisi tavanomaisen hinnoittelun mukaan maksanut 5 000 dollaria, sillä Mevean viime vuoden liikevaihto oli 1 270 000 euroa (10). Tämä olisi kuitenkin muuttunut, sillä FMOD:a olisi käytetty simulaattoreissa. Myös FMOD:n käyttö useammassa eri simulaattorissa olisi mahdollisesti nostanut hintaa vielä korkeammaksi, eikä tähän sijoitukseen haluttu panostaa, ainakaan vielä tämän projektin aikana. Lopputuloksena päätettiin käyttää Unityn omaa äänimoottoria.

Projektissa käytettiin myös erillistä DAWia eli digitaalista äänityöasemaa, jolla muokattiin äänitiedostoja tarpeen mukaan. REAPER on Cockosin valmistama äänityöasema, jota käytettiin tässä projektissa, koska siihen löytyi jo henkilökohtainen lisenssi.

Digitaalisella äänityöasemalla (DAW) tarkoitetaan elektronista laitetta tai sovellusta, jota käytetään äänitiedostojen tallentamiseen, muokkaamiseen ja tuottamiseen. DAW-laitteiksi voidaan laskea monenlaisia kokoonpanoja yhdestä kannettavan tietokoneen ohjelmistosta yksinkertaiseen kotistudioyksikköön (esim. mikseri, äänikortti ja tietokone) ja aina erittäin monimutkaiseen kokoonpanoon, joka koostuu useista keskustietokoneen ohjaamista komponenteista. Kokoonpanosta riippumatta nykyaikaisissa DAW-asemissa on keskitetty käyttöliittymä, jonka avulla käyttäjä voi muokata ja miksata useita tallenteita ja raitoja lopulliseksi tuotokseksi. (11.)

4 Käyttöliittymän ja äänimaailman toteutus

Projektin kulku kuvataan insinöörityöraportissa kahdessa osassa, ensin HMI-ruutu, sitten äänet. Todellisuudessa molempia projekteja tehtiin päällekkäin, vuoroviikoin. Aikataulullisesti äänimaailman suunnitteluun ja tekemiseen varattiin enemmän aikaa, sillä HMI-ruudun suunnittelu ja kehitystyö oli jo aloitettu. Molemmille osille oli varattu 7–8 viikkoa aikaa, mukaan lukien suunnittelu- ja tutkimustyö.

Projekti suoritettiin täysin etätyöskentelynä. Joka viikon lopuksi pidettiin palaveri, jossa seurattiin projektin etenemistä. Palaverit toimivat myös hyvinä välietappeina, joissa vaihdettiin toteutusideoita ja voitiin esim. karsia turhia ominaisuuksia pois. Pääasiallisina viestintäkanavina toimivat Slack ja Microsoft Teams.

Projektin molempien osien perusideana oli, että kaikkien muutoksien ja uusien ominaisuuksien haluttiin olevan niin yksinkertaisia kuin mahdollista, jotta niitä voidaan helposti lisätä eri simulaattoreihin. Tämä helpotti myös testausta melko tiukassa ja lyhyessä aikataulussa.

4.1 Käyttöliittymä

HMI-projekti eli simulaattorin käyttöliittymän parantaminen aloitettiin palaverilla, jossa kartoitettiin projektin tarpeet ja aloitettiin suunnittelutyötä Mevean jo olemassa olevilla dokumentaatioilla. Niihin kuului esimerkiksi valmiiksi toteutettuja konsoleita ja mittaristoja eri simulaattoreista, muutamia kuvia ja videoita eri valmistajien pyöräkuormaimien ja kaivinkoneiden hallintalaitteistoista. Tämän lisäksi käytiin myös läpi vanhan HMI-projektin kansioita ja tiedostoja. Tämän pohjalta selvitettiin, mitä komponentteja voisi käyttää uudessa projektissa, ja niitä kerättiin PowerPoint-esitykseen. Se sisälsi esimerkiksi isoissa satamanostureissa käytettäviä nappeja, jotka olivat todella laadukkaita. Mevea oli valmistanut ne asiakasprojektia varten, mutta niitä voitiin käyttää hyväksi myös tässä projektissa.

Tämän lisäksi haluttiin selvittää, mitä kaikkea uutta valmistajilla on nykyään tarjolla ja mitkä kaikki tarvittavat mittarit ja ominaisuudet ovat vakioina uusissa työkoneissa. Tätä varten tutkittiin noin kuuden eri valmistajan eriluokkaisia kaivinkoneita ja kerättiin niistä talteen kaikki mittaristot ja hallintalaitteet, jotka sijoitettiin yhteen PowerPoint-esitykseen. PowerPoint-esitys käytiin läpi palaverissa. Palaverissa päätettiin, mitä mittareita haluttiin käyttää, ja puhuttiin myös mahdollisista toteutuksista.

Yksi HMI-projektin ongelma oli sen epäselvyys. Mitään vanhoja tiedostoja ei ollut koskaan siivottu pois, mikä hankaloitti oikeiden komponenttien ja tietojen etsimistä. Projektin kansiorakenteet olivat myös melko epäselvä, mikä hankaloitti projektin aloitusta entisestään. Ensimmäinen versio projektista tehtiinkin eri projektikansioon, mutta tästä ilmeni ongelmia, joiden ratkomiseen ei ollut käytettävissä tai ei haluttu käyttää aikaa. Yksi ongelmista olivat Mevea Software API:n liitännät, joita käyttivät useat eri mittari- ja nappikomponentit. Niiden siirtäminen ja toiminnan tarkastaminen olisi vienyt paljon aikaa, koska insinöörityön tekijällä ei ollut kokemusta Mevean ohjelmista. Tästä syystä päätettiin jatkaa vanhalla projektilla, johon tehtiin uusi kansio, minne lisätiin kaikki mahdolliset muutokset.

HMI-ruutuun tarvittiin neljä eri näkymää: mittarit, napit, jatkokehitystä varten kamera sekä mittareiden ja nappien konfigurointi. Kaikki eri näkymät ovat erillisiä canvas-elementtejä, jotka voidaan sijoittaa eri näytöille, jos esimerkiksi simulaattorissa on käytössä monta eri fyysistä näyttöä. Napit on koodattu niin, että ne etsitään ensin nimellä, minkä jälkeen jokaiseen nappiin asetetaan kuuntelija, joka kuuntelee funktiota. Tämän funktion (esimerkkikoodi 1) tehtävänä on deaktivoida muut näkymät ja asettaa painettavan napin nimellä varustettu näkymä aktiiviseksi.

```
Unity Message | 0 references
void Start()
{
    Cauge = GameObject.Find("Cauges_Canvas").GetComponent<Canvas>();
    Cauge.gameObject.SetActive(true);
    Button ca_btn = Cauge_btn.GetComponent<Button>();
    ca_btn.onClick.AddListener(CaugeActive);

    Button = GameObject.Find("Buttons_Canvas").GetComponent<Canvas>();
    Button.gameObject.SetActive(false);
    Button b_btn = Button_btn.GetComponent<Button>();
    b_btn.onClick.AddListener(ButtonActive);

    Camera = GameObject.Find("Camera_Canvas").GetComponent<Canvas>();
    Camera.gameObject.SetActive(false);
    Button camera_btn = Camera_btn.GetComponent<Button>();
    camera_btn.onClick.AddListener(CameraActive);

    Config = GameObject.Find("Config_Canvas").GetComponent<Canvas>();
    Config.gameObject.SetActive(false);
    Button co_btn = Config_btn.GetComponent<Button>();
    co_btn.onClick.AddListener(ConfigActive);
}

1 reference
void CaugeActive()
{
    Camera.gameObject.SetActive(false);
    Config.gameObject.SetActive(false);
    Cauge.gameObject.SetActive(true);
    Button.gameObject.SetActive(false);
}
```

Esimerkkikoodi 1. HMI-näkymien vaihdosta vastaava koodinpätkä.

Konfiguraationäkymä ei saa olla näkyvissä simulaattorin ajon aikana. Helpoin tapa toteuttaa tämä oli piilottaa eri näkymiä vaihtavat napit. Tätä varten lisättiin CTRL+Z-näppäinyhdistelmä (esimerkkikoodi 2), ettei simulaattorin ohjaaja tai käyttäjä paina vahingossa mitään yksittäistä näppäintä.

```
void Start()
{
    Config = GameObject.Find("Config_Buttons");
    Config.gameObject.SetActive(false);
}

// Update is called once per frame
Unity Message | 0 references
void Update()
{
    if ((Input.GetKey(KeyCode.RightControl) || Input.GetKey(KeyCode.LeftControl)) && Input.GetKeyDown(KeyCode.Z))
    {
        if (Config.activeSelf)
        {
            Config.gameObject.SetActive(false);
        }
        else
        {
            Config.gameObject.SetActive(true);
        }

    }
}
```

Esimerkkikoodi 2. Näkymien piilotuksesta vastaava koodinpätkä.

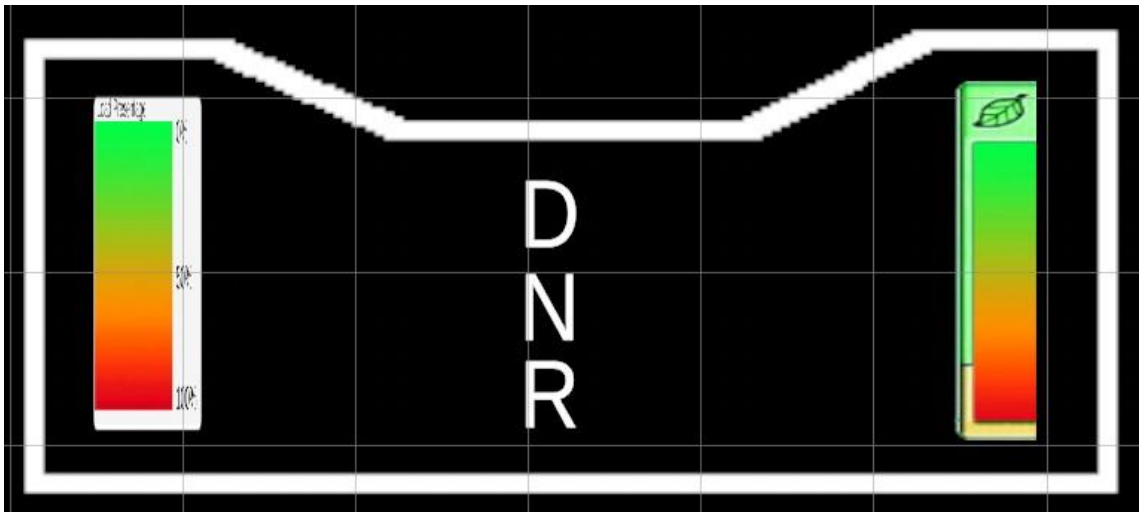
Yksi tärkeimmistä pohjan ominaisuuksista sen joustavuuden, monipuolisuuden ja jatkokehityksen kannalta oli mahdollisuus vaihtaa mittareita riippuen käytettävästä simulaattorista. Tässä projektissa keskityttiin vain kaivinkoneen kannalta oleellisiin komponentteihin. Jotta pohjaa voidaan käyttää eri simulaattoreissa, joissa mittarit ovat erilaisia tai joitain mittareita halutaan priorisoida, mittareita piti pystyä vaihtamaan helposti.

Toteutus (esimerkkikoodi 3) oli yksinkertainen, mutta se täyttää kaikki asetetut vaatimukset. Mittarit ja napit lisättiin listaan, joka lisätään pudotusvalikkoon. Pudotusvalikosta valittu komponentti aktivoidaan ruudulla ja muut mittarit deaktivoidaan for-silmukassa. Komponentit on asetettu ruudulla päällekkäin. Vakioasetuksena mittarin paikka on merkitty kirjaimella, jota vastaa konfigurointinäkymässä lista samalla kirjaimella. Näin mittarien paikat on helppo havainnollistaa.

```
Unity Script (19 asset references) | 0 references
public class Drop : MonoBehaviour
{
    public Dropdown dropdown;

    public GameObject[] buttons;

    Unity Message | 0 references
    private void Start()
    {
        dropdown.onValueChanged.AddListener(delegate
        {
            changebusdf(dropdown);
        });
    }


    1 reference
    public void changebusdf(Dropdown yes)
    {
        for (int i = 0; i < buttons.Length; i++)
        {
            buttons[i].SetActive(false);
            buttons[yes.value].SetActive(true);
        }

    }
}
```

Esimerkkikoodi 3. Mittarien listauksesta vastaava koodinpätkä.

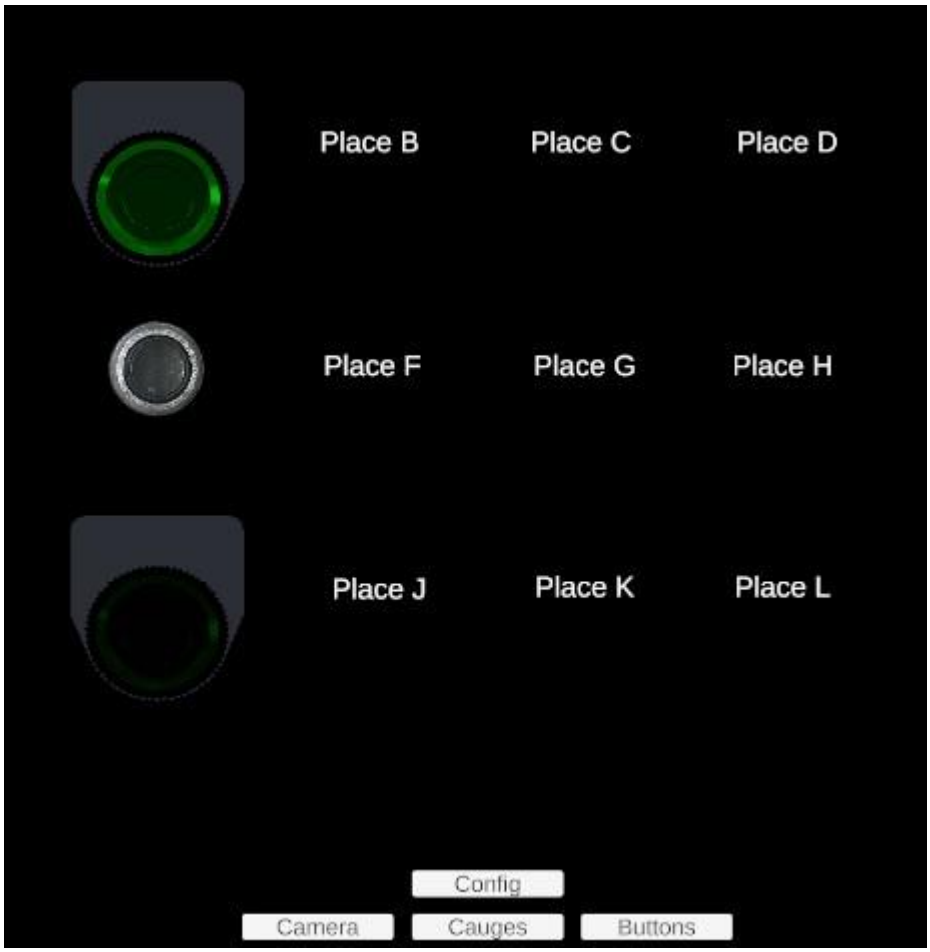
Tämän projektin aikana oli tarkoitus saada Mevealta uusia mittareita ja lisätä Unityn vakiovärin päälle uusi pohja. Tähän tehtävään tarkoitettu Mevealla myös työharjoittelussa oleva opiskelija ei ehtinyt sitä tekemään suunnitellussa aikataulussa. Tämän takia jouduttiin tekemään itse uusi pylväsmittari (kuva 5), josta tehtiin kaksi eri versiota: moottorin ekologisuusmittari ja kauhan maahan kohdistuvan paineen mittari, vaihdevalot ja uusi pohja. Pohja toteutettiin GIMP-valonkuvakäsittelyohjelmassa ja uudet mittarit Unityssä. Muut mittarit ovat Mevean tekemiä, ja niitä on käytetty muissa asiakasprojekteissa. Niitä jouduttiin hieman hienosäätämään mm. vaihtamalla mittareiden väriä.



Kuva 5. Valmis mittariston keskiosa, jossa kaksi uutta pylväsmittaria ja vaihteenvalitsin.

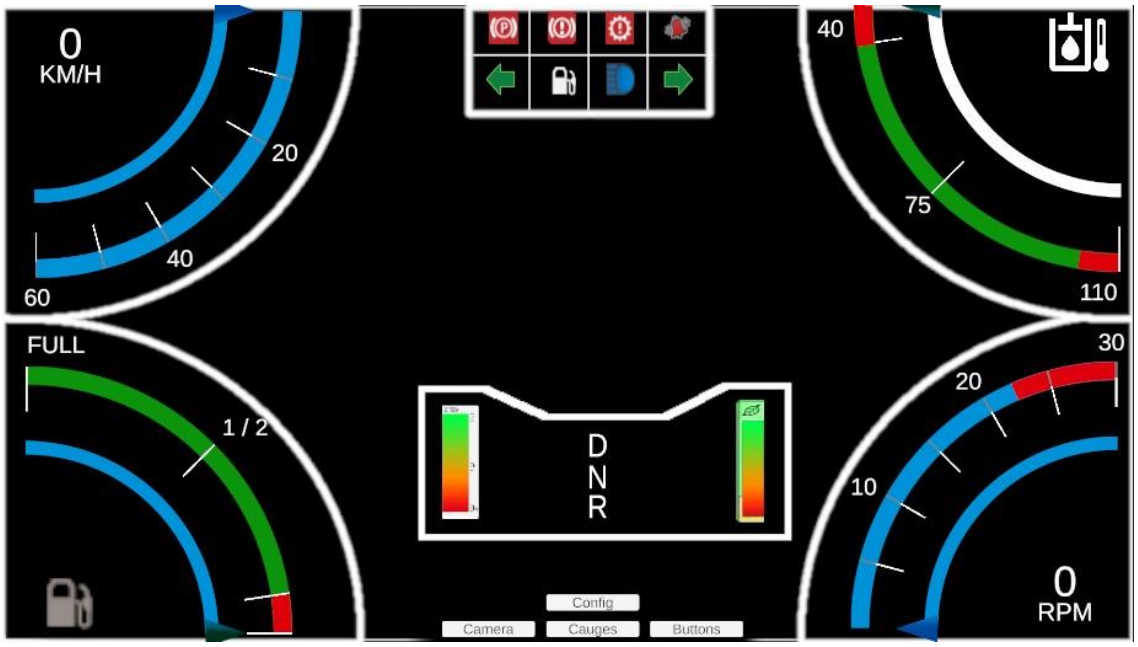
Pylväsmittarin suunnittelu oli nopea prosessi, sillä ainoat mittarin vaatimukset olivat sen malli ja tarvittavat mittausyksiköt mittarin sivulla. Molemmissa mittareissa arvo esitetään väriradiantilla, joka on punainen, kun mitattava arvo on maksimi, ja vihreä, kun arvo on minimi. Näitä mittareita ei ollut tarkoitus tehdä toimiviksi, sillä teknistä toteutusta ei vielä Mevean puolelta ollut. Mittarin arvoa pystyy kuitenkin muuttamaan ja lukemaan funktioilla, joihin voi lopuksi lisätä koodia, jotta mittarit saadaan toimimaan simulaattorissa.

Nappien puolesta muutoksia innovaatioprojektista ei ollut paljoa. Taustan väri muutettiin mustaksi ja lisättiin yksi uusi nappi testausta varten. Napeille ei vielä asetettu mitään toiminnallisuutta. Napit (kuva 6) ovat Mevean muista simulaattoreista tuotuja pääasiassa testausta varten. Uusia nappejakin oli vielä suunnitteluvaiheessa tarkoitus lisätä, mutta niistäkin luovuttiin aikataulun takia.



Kuva 6. Nappien näkymä, jossa esillä kolme esimerkkinappia. Alakulmassa napit eri näkymille.

Valmiissa mittarinäkymässä (kuva 7) on paikka neljälle puolikaaren muotiselle mittarille, kahdelle pylväsmittarille, vaihteenvalitsimelle ja kahdeksalle merkkivalolle. Merkkivalot ovat toistaiseksi vain paikanvaraajia, eikä niihin haluttu käyttää aikaa. Kameranäkymä on poista vain kuvankaappaus Komatsun PC-sarjan kaivinkoneen HMI-näytöstä.



Kuva 7. Valmis mittariston näkymä.

Pohjan valkoiset alueet tehtiin GIMP-kuvanmuokkaustyökalulla.

4.2 Äänimaailma

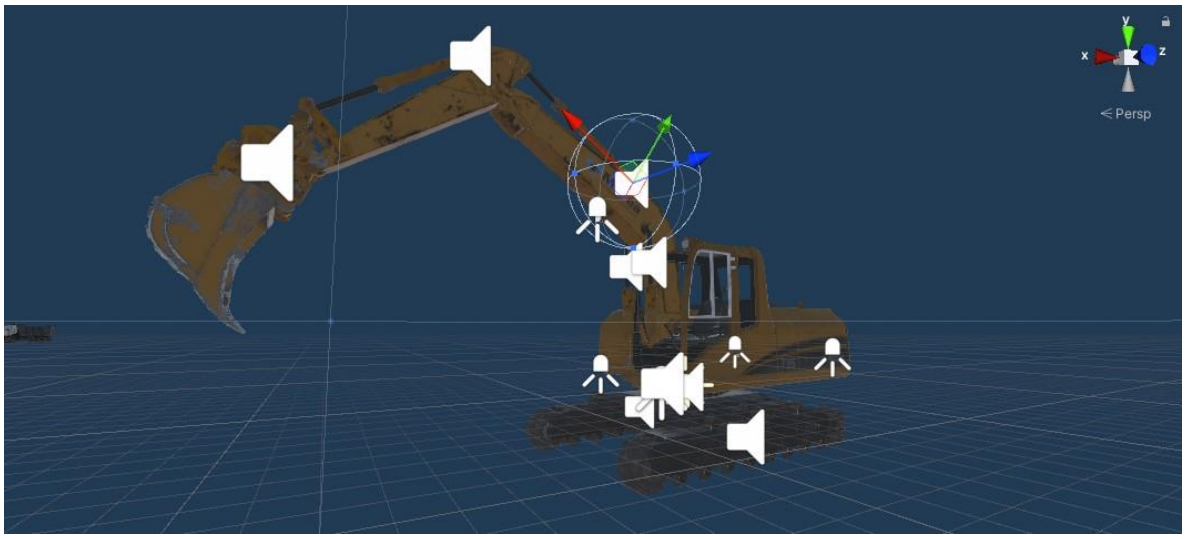
Äänimaailman luomisen ensimmäiset vaiheet olivat tarvittavien äänikirjastojen etsiminen ja niiden käytön suunnitteleminen. Projektia varten kerättiin internetistä eri äänikirjastoja, osa maksullisia ja osa ilmaisia. Niihin lukeutui muun muassa eri työmaiden työkoneiden ääniä, satama-aiheisia taustaääniä ja kaivinkoneiden eri sylintereiden ja moottoreiden ääniä. Nämä kaikki kerättiin yhteen PowerPoint-esitykseen, jossa käytiin myös läpi äänimoottorin valinta. Kun äänikirjastoja oli kerätty tarpeeksi, kaikki kirjastot käytiin läpi yhdessä, jolloin kuunneltiin äänileikkeitä kaikista esitykseen kerätyistä äänikirjastoista. Lopuksi niistä valittiin parhaat, jotka arkistoitiin myös Mevean omille palvelimille. Projektissa käytettäväksi jäljelle jäi yksi maksullinen kirjasto ja kolme eri ilmaista kirjastoa.

4.2.1 Kaivinkoneen äänet

Kaivinkoneen äänien suunnittelu alkoi vanhojen äänien ja projektin tarkastelulla. Tarkoitus oli kartoittaa projektin eri ääniä ohjaavia koodeja ja sitä, mitä ääniä voi

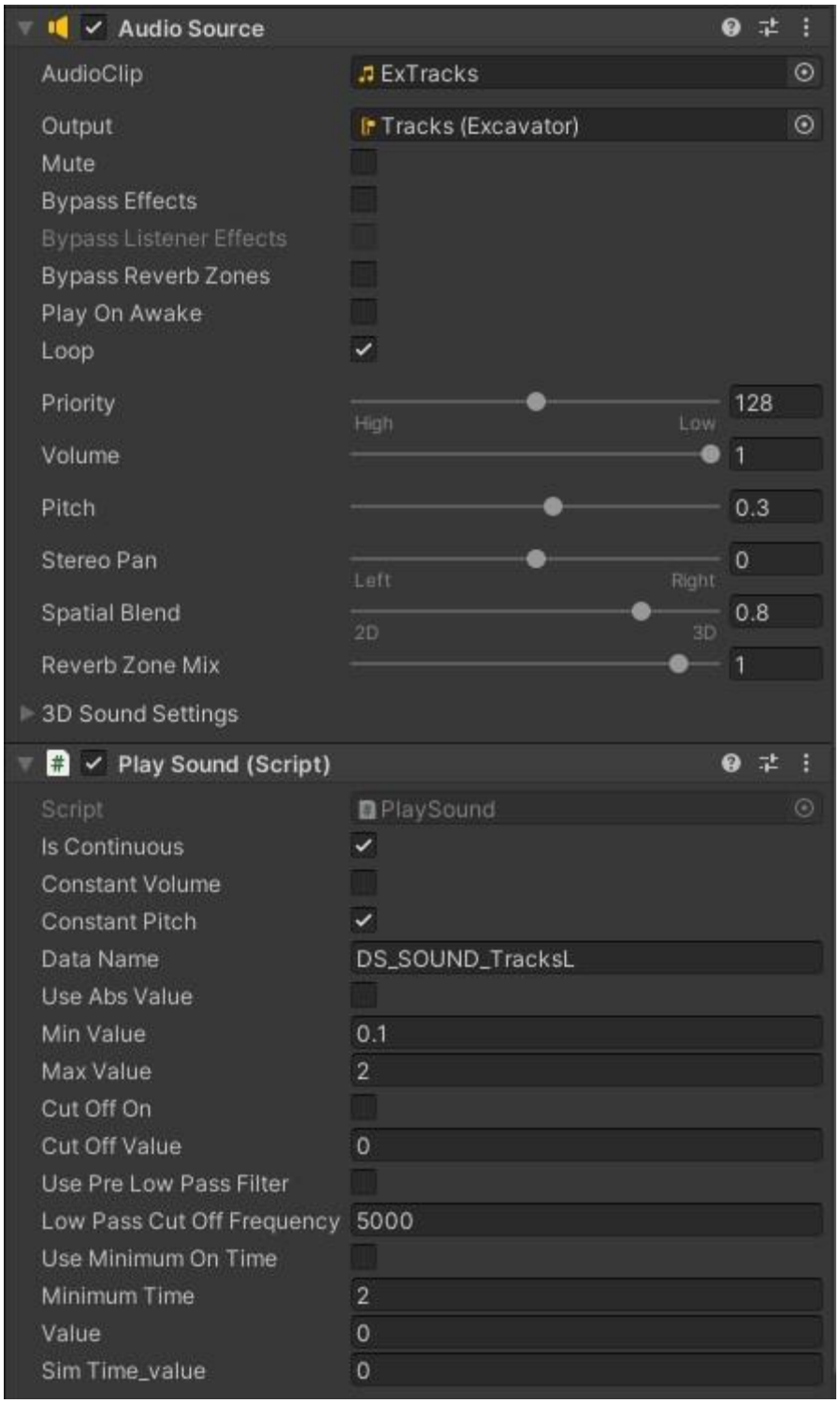
ns. vain vaihtaa ja mitä piti lisätä. Kaivinkoneessa oli valmiiksi jo kaikki tarvittavat komponentit, eli tehtäväksi jäi vain vaihtaa olemassa olevat äänet uusiin ja muokata tarvittaessa ääniä soittavaa koodia.

Kaivinkoneen mallissa (kuva 8) on useita eri äänikomponentteja. Kauhan ja varren kolme ylimmäistä komponenttia toistavat hydraulipumpun eri ääniä. Kaksi muuta kopin vieressä puomiin kiinnitettynä olevaa komponenttia toistavat raskaamman työkoneen hydraulipumppua, joka on eri kirjastosta kuin ylemmät. Näiden komponenttien äänentaajuutta säädellään koodissa riippuen siitä, kuinka nopeasti vartta liikutetaan ja kuinka paljon rasitusta puomiin kohdistuu. Kopin edessä oleva komponentti on äänitorvi, ja sen takana on moottorin äänikomponentti. Molemmilla telaketjuilla on myös omat äänikomponentit.



Kuva 8. Äänikomponenttien paikat kaivinkoneen mallissa.

Jokaisessa komponentissa on liitettynä äänilähde, joka soittaa haluttua ääntä, ja PlaySound-koodi. Äänilähteessä (kuva 9) on tärkeää, että silmukkaominaisuus on laitettu päälle, mikäli se on äänelle oleellista. PlaySound-koodissa on oma IsContinuous-ominaisuus, mutta se on tarkoitettu vain ohjamaan ääntä koodissa.



Kuva 9. Toisen telaketjun äänikomponentin rakenne, joka toistuu myös muissa äänikomponenteissa.

Vaikka äänilähteissä on kaikissa oma äänenvoimakkuuden säätö ja esimerkiksi äänen korkeuden säätö, kaikki äänilähteet on asetettu vakioarvoilla. Tämä helpotti äänien miksausta, joka tehtiin audiotyöasemassa. Näin kaikkia ääniä pystyi

kokeilemaan samaan aikaan, ja ne oli helppo miksata. Miksauksen jälkeen kaikki äänileikkeet lisättiin oikeisiin komponentteihin.

PlaySound-koodissa on monta eri osaa, joilla on useita eri tarkoituksia, varsinkin kun sitä käytetään myös esimerkiksi kippirekan äänissä. Oleellinen osa kaivinkoneen äänien kannalta on PlayAudio-niminen funktio, joka ottaa vastaan liukuluvun audioValue. Funktion tarkoituksena on toistaa eri äänitiedostoja. Koodi ei kuitenkaan toiminut suoraan yhteen tehtyjen muutoksien kanssa, joten sitä piti muokata. Toistokäyttäytyminen perustuu audioValue-parametrin arvoon ja muutamaan muuhun ehtoon. Funktiossa on eri if-lauseita, jotka asettavat ehtoja, miten ja milloin ääntä pitää soittaa.

Seuraavan funktion erittelyn esittämisen helpottamiseksi funktion eri muuttujat listattiin taulukkoon, sitä mukaa, kuin niitä käytettiin koodissa. Termi totuusarvomuuttuja (Boolean muuttuja) tarkoittaa tulosta, joka voi olla vain toinen kahdesta mahdollisesta arvosta: tosi tai epätosi. Liukuluku taas tarkoittaa tapaa esittää luku arvon ja suuruusluokan yhdistelmänä.

Taulukko 1. Funktion muuttujien toimintaperiaatteet.

Muuttuja	**Toimintaperiaate**	**Totuusarvomuuttuja**
doPlay	Jos tosi, soitetaan äänileike	Totuusarvomuuttuja
m_minValue	Tarkastaa audioValuen arvon	Liukuluku
m_maxValue	Tarkastaa audioValuen arvon	Liukuluku
_hasBeenPlayed	Jos tosi, ääni soitettu	Totuusarvomuuttuja
m_useMinimumOnTime	Toistaa äänen vain, jos signaali on täyttänyt ehdot	Totuusarvomuuttuja

Muuttuja	**Toimintaperiaate**	**Totuusarvomuuttuja**
_isWaiting	Toimii välivaiheena ennen äänen soittamista	Totuusarvomuuttuja
_startTime	Tarvitaan äänen soittamiseen oikeaan aikaan	Liukuluku

Funktio (esimerkkikoodi 4) toimii eriteltynä niin, että aluksi var-muuttuja doPlay alustetaan epätodella arvolla. Seuraavaksi on if-lause, joka tarkistaa, onko m_minValue pienempi kuin m_maxValue. Jos tämä on tosi ja audioValue on suurempi kuin m_minValue ja pienempi tai yhtä suuri kuin m_maxValue, doPlay-muuttuja asetetaan todeksi. Mikäli m_minValue on suurempi tai yhtä suuri kuin m_maxValue, koodi siirtyy else-lauseeseen, jonka sisällä on if-lause, joka tarkistaa, onko audioValue suurempi kuin m_minValue tai pienempi tai yhtä suuri kuin m_maxValue. Mikäli nämä ehdot täyttyvät, myös tämän lauseen jälkeen doPlay-muuttuja on tosi.

Koodin seuraava osa tarkastaa, ettei tiettyä ääntä soi päällekkäin useita kertoja. If-lause tarkistaa ensin, onko ääntä asetettu silmukalle. Mikäli ääni ei ole asetettu silmukalle, menee koodi eteenpäin seuraavaan if-lauseeseen, joka katsoo, onko doPlay tosi. Seuraava lause tekee saman _hasBeenPlayed-muuttujalle. Mikäli molemmat ehdot ovat tosi, doPlay asetetaan arvoon epätosi. Jos doPlay on kuitenkin epätosi, _hasBeenPlayed asetetaan arvoon epätosi.

Seuraava if-lause tarkistaa, ovatko doPlay ja m_useMinimumOnTime tosia. Jos molemmat ovat tosia ja ääntä ei toisteta eikä _isWaiting ole tosi, _startTime asetetaan nykyiseen aikaan ja _isWaiting on tosi. Jos ero nykyisen ajan ja _startTimen välillä on pienempi tai yhtä suuri kuin m_minimumTime, doPlay asetetaan arvoon epätosi. Mikäli ensimmäisen if-lauseen ehdot eivät täyty, _isWaiting on epätosi, mikä tarkoittaa, että ääntä voi soittaa.

Jos doPlay on edelleen tosi, seuraava If-lause tarkistaa, onko ääni asetettu silmukalle. Jos on ja ääntä ei juuri toisteta, se toistetaan. Jos sitä ei ole asetettu silmukaksi, ääni toistetaan kerran käyttäen PlayOneShot-menetelmää ja _hasBeenPlayed muutetaan todeksi, etteivät ylemmät lauseet soita ääntä uudestaan. Mikäli doPlay on kuitenkin epätosi, ääni on asetettu silmukalle ja sitä toistetaan parhaillaan, se pysäytetään. Kaiken kaikkiaan PlayAudio-funktio on menetelmä, jolla voidaan toistaa äänitiedostoa ja jolla on ehdollinen toistokäyttäytyminen, joka perustuu arvoalueeseen (m_minValue - m_maxValue), onko ääni asetettu silmukaan vai ei, ja vähimmäisaika-arvoon (m_minimumTime).

```
private void PlayAudio(float audioValue)
{
    var doPlay = false;

    if (m_minValue < m_maxValue)
    {
        if (audioValue > m_minValue && audioValue <= m_maxValue)
            doPlay = true;
    }
    else
    {
        if (audioValue > m_minValue || audioValue <= m_maxValue)
            doPlay = true;
    }

    if (!_audioData.loop)
    {
        if (doPlay)
        {
            if (_hasBeenPlayed)
                doPlay = false;
        }
        else
        {
            _hasBeenPlayed = false;
        }
    }

    if (doPlay && m_useMinimumOnTime)
    {
        if (!_audioData.isPlaying && !_isWaiting)
        {
            _startTime = System.DateTimeOffset.Now;
            _isWaiting = true;
        }

        if ((System.DateTimeOffset.Now - _startTime).TotalMilliseconds / 1000
            <= m_minimumTime)
        {
            doPlay = false;
        }
    }
    else
    {
        _isWaiting = false;
    }

    if (doPlay)
    {
        if (_audioData.loop)
        {
            if (_audioData.isPlaying == false)
                _audioData.Play();
        }
        else
        {
            //Debug.Log("Playing");
            _audioData.PlayOneShot(_audioData.clip, 1.0f);
            _hasBeenPlayed = true;
        }
    }
    else
    {
        if (_audioData.loop)
        {
            if (_audioData.isPlaying)
                _audioData.Stop();
        }
    }
    //Debug.Log("Value " + audioValue);
}
```

Esimerkkikoodi 4. PlayAudio-funktio.

Funktiossa on myös Depug.log-komentoja, jotka auttoivat paljon koodia muokattaessa. Ne jätettiin koodiin siksi, että seuraavan, joka koodia tarvittaessa muokkaa, on helpompi ymmärtää, mitä mikäkin lause tekee.

4.2.2 Taustaäänet

Taustaäänien suunnittelussa perehdyttiin etenkin satamassa kuuluviin ääniin. Niihin kuuluivat eri työkoneiden äänet, satamassa kuuluvat mahdollisesti eri eläinten äänet (lokit, hylkeet yms.) ja myös satamassa operoivat laivat. Työkoneiden äänillä haluttiin kasvattaa vaikutelmaa, että kaivinkone työskentelisi vilkkaalla työmaalla. Koska simulaattorin ympäristö on rakennettu satama-alueelle, haluttiin sinne myös visuaalista puolta myötäilevä äänimaailma.

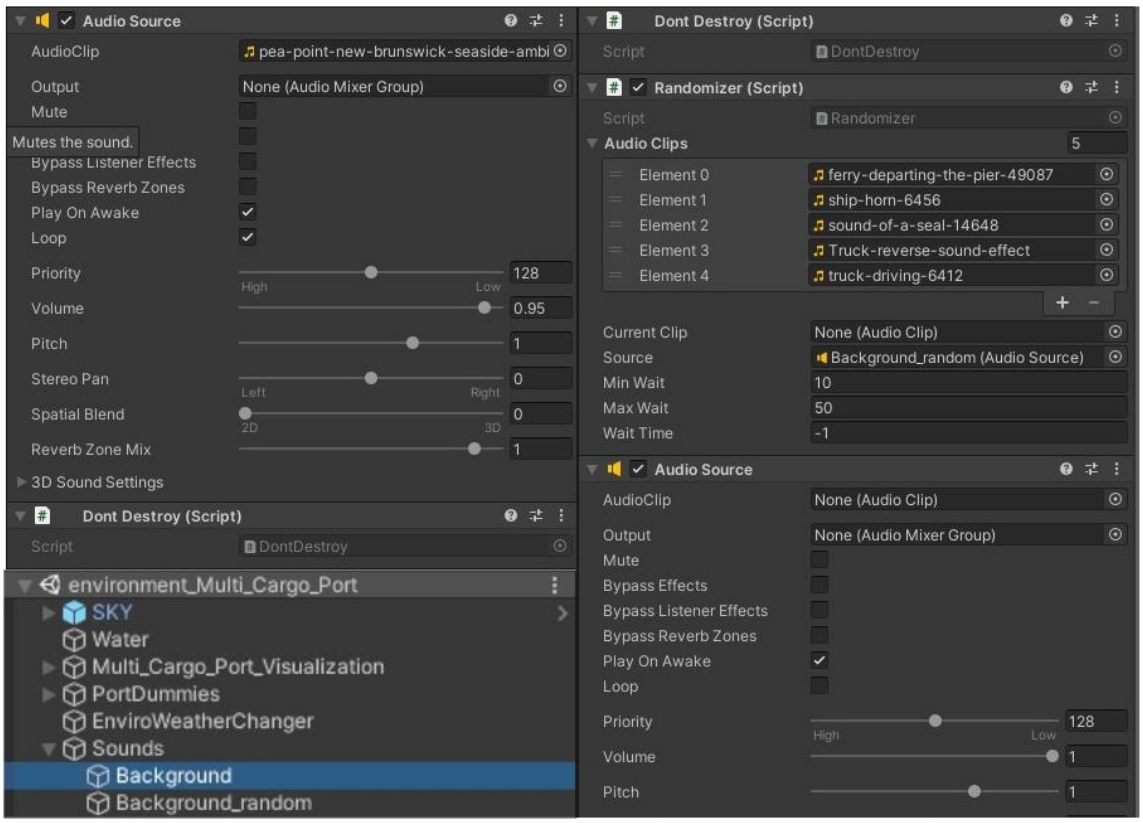
Aluksi mietittiin eri äänien äänittämistä esimerkiksi Vuosaaren satamassa, joka vastaa melko lailla Mevean rakentamaa simulaatioympäristöä. Tämä olisi kuitenkin vienyt liikaa aikaa ja internetistä löytyi jo valmiiksi tarpeeksi hyviä audioleikkeitä, joita voitiin käyttää.

Koska ääniä on monta erilaista eikä niitä kaikkia haluttu soittaa samanaikaisesti, tarvittiin koodi, joka soittaa eri ääniä täysin satunnaisesti. Koodissa (esimerkkikoodi 5) on Unityn oma päivitysfunktio, jota kutsutaan jokaisella ruudunpäivityksellä. Mikä tahansa päivitystoiminnon sisällä oleva komentosarja suoritetaan kerran jokaisessa ruudunpäivityksessä. Eri äänille tehdään lista, josta koodin päivityslause käy satunnaisesti poimimassa aina yhden soitettavan äänen. Päivityslausessa on myös waitTime-luku, joka saadaan, kun koodi valitsee kahden muun luvun, minWait ja maxWait, väliltä satunnaisesti jonkin luvun. Esimerkiksi jos minWaitin arvoksi on asetettu 10 sekuntia ja maxWaitin 30 sekuntia, waitTime voi olla mikä tahansa sekuntiluku näiden kahden väliltä. Tulos ei ole myöskään tasaluku, vaan voi olla esimerkiksi 12,52 sekuntia. Tällä ei ole tämän koodin käyttötarkoituksessa mitään väliä.

```
public class Randomizer : MonoBehaviour
{
    public List<AudioClip> audioClips;
    public AudioClip currentClip;
    public AudioSource source;
    public float minWait = 1f;
    public float maxWait = 5f;
    public float waitTime = -1f;

    Unity Message | 0 references
    void Start()
    {
        source = GetComponent<AudioSource>();
    }

    Unity Message | 0 references
    void Update()
    {
        if (!source.isPlaying)
        {
            if (waitTime < 0f)
            {
                currentClip = audioClips[Random.Range(0, audioClips.Count)];
                source.clip = currentClip;
                source.Play();
                waitTime = Random.Range(minWait, maxWait);
            }
            else
            {
                waitTime -= Time.deltaTime;
            }
        }
    }
}
```

Esimerkkikoodi 5. Randomizer-koodi, joka soittaa satunnaisesti eri ääniä.

Taustalla soitetaan jatkuvasti yhtä seitsemän minuutin pituista taustaääntä, jonka päälle on lisätty satunnaisesti soitettavat äänet. Molemmat eri taustaäänet ovat eri peliobjekteja (kuva 10), joissa on molemmissa oma audiolähde. Molemmat äänilähteet on miksattu stereona, mutta satunnaisesti soitettavat äänileikkeet on miksattu erikseen audiotyöasemassa.



Kuva 5. Taustaäänien komponenttien rakenne.

Yhdessä viikkopalaverissa keskusteltiin taustaäänien mahdollisista efekteistä. Simulaattorin tai kaivinkoneen ajon aikana käyttäjä luonnollisesti istuu suljetussa ympäristössä, jolloin ääni kuuluu eri tavalla kuin esimerkiksi suoraan ilman efektejä soitettuna äänitiedostona, joka on äänitetty ulkoilmassa. Tätä suljetun tilan efektiä testattiin erityyppisillä kaiuilla ja äänentaajuuksien muutoksilla, mutta ne eivät tuottaneet haluttua lopputulosta.

Taustaääniä soitetaan kuitenkin melko hiljaisella verrattuna esimerkiksi koneen äänin, joten huomattavaa eroa ei edes syntynyt. Taustaäänet toimivat myös parhaiten tässä tilanteessa, kun ne katoavat taka-alalle silloin, kun simulaattoria käytetään, ja nousevat takaisin esiin, kun muu työkoneen käytöstä kuuluva ääni loppuu. Lopputuloksena on juuri suunnitteluvaiheessa haluttu immersion tunteen lisääminen.

5 Projektin lopputulos

Projektissa moni asia onnistui erinomaisesti, mutta joitakin asioita jäi myös tekemättä, lähinnä aikarajoitteiden ja 3D-malleihin erikoistuneen osaajan puuttumisen vuoksi. Tuloksena oli kuitenkin toimiva HMI-pohja, jota jatkokehitetään edelleen ja jonka lisäämisestä kaikkiin Mevean omiin simulaattoreihin keskustellaan jatkossa. Myös uudet äänet saatiin toteutettua suurimmalta osin. Projektin molemmat osat saatiin aikataulun puitteissa valmiiksi.

Yhteistyö sujui projektin kaikkien osapuolien kanssa oikein hyvin. Ideoiden esittäminen ja vaihtelu oli todella helppoa ja vaivatonta. Oman osaamiseni pohjalta pystyin helposti sanomaan, mihin pystyin ja mihin en, ilman että se olisi tuntunut jotenkin tuomittavalta. Kaikki projektiin osallistuvat olivat avuliaita ja tarpeeksi vaativia. Mielipiteitä pystyttiin kuuntelemaan puolin ja toisin.

5.1 Käyttöliittymä

Tärkeimpiä tämän osan saavutuksia olivat HMI-pohjan valmiiksi saanti testausta varten aikataulun puitteissa ja sen osittainen testaus. Alun perin myös ns. lopputestaus oikealla simulaattorilla (kuva 11) kuului projektin tavoitteisiin, mutta siitä luovuttiin pääosin sen takia, että testaukseen vaadittiin fyysinen simulaattori, johon on liitettynä yksi tai useampia HMI-näyttöjä. Tässä projektissa siis riitti, että mittarien arvoja pystyy muuttamaan Unityssä ja että luvut näkyvät oikein. Uudet äänet saatiin testattua helposti etänä, sillä simulaattoria pystyy ohjaamaan helposti virtuaalisesti Unityssä Solver-sovelluksen avulla.

Kuva 6. Mevean kaivinkonesimulaattori käytössä.

Projektin isoin ongelma oli uusien mittarien puute ja mittarinäkymän taustan viimeistely. Taustaan olisi tarvinnut kiinnittää enemmän huomiota ja käyttää siinä ammattitaitoista osaamista. Myös vanhat mittarit olivat hieman vanhanaikaisia eivätkä täyttäneet alkuperäisiä suunnitelmia. Graafikoita ei valitettavasti tämän projektin aikataulutuksen takia saatu auttamaan näissä asioissa. Tämä ei Mevealla haitannut, mutta insinöörityön tekijän näkökulmasta lopputulos olisi ollut parempi ja lähempänä nykyaikaista ilmettä, mitä alun perin haettiin.

Mittarinäkymän pohjan puolikaarista ei saatu täysin symmetrisiä, eivätkä keskellä olevat mittarit ole täysin keskellä verrattuna reunoihin. Tämä ei kuitenkaan vaikuta mittarin toiminnallisuuteen, eikä aikaa ollut enää jäljellä sen parantamiseen.

5.2 Äänet

Kaikki muut suunnitellut äänet saatiin lisättyä paitsi kaivamiseen liittyvät maansiirtoäänet. Ne osoittautuivat haastavaksi koodata, myös osittain Unityn oman äänimoottorin teknisten rajoitusten takia. Kaikki yritykset, jotka saatiin toimivaksi, osoittautuivat nopeasti äänimaailmaa ja samalla immersion tunnetta rikkoviksi, ja immersiohan oli koko projektin pääidea.

Taustaäänien lisäys sai todella hyvän vastaanoton. Mevean tekemissä testeissä lähes kaikki käyttäjät sanoivat, että ero vanhaan on huomattava. Tämä tuli hieman yllätyksenä, sillä vaikka taustaäänien puutteesta oli paljonkin puhetta projektin suunnitteluvaiheessa, taustaäänien tarkoitus on olla taka-alalla, kun simulaattoria käytetään, ja tulla esiin vain silloin, kun esimerkiksi moottori on sammuksissa. Tämä on kuitenkin immersion kannalta tärkeää, sillä oikeassa työmaaympäristössä on aina eri koneiden ääniä ja muita, esimerkiksi luonnon ääniä. Tämän simulointiin toimivat erityisen hyvin lisätyt satunnaiset äänet. Ne lisäävät hyvin immersion tunnetta ja auttavat simulaattorin käyttäjää opettelemaan ottamaan huomioon myös työmaa-alueella kulkevat ajoneuvot. Tästä hyvänä esimerkkinä on kuorma-auton liikkeellelähdön ääni, joka soi satunnaisesti simulaattorin ollessa käynnissä. Taustaäänien puuttumiseen ja niiden vaikutukseen ei ollut aiemmin kiinnitetty huomiota, paljolti sen vuoksi, että Mevealla ei ole varsinaista ääniin erikoistunutta työntekijää.

5.3 Tulevaisuus

Projektin tulevaisuudesta oli yrityksessä puhetta. Uudet mittarit ja napit ovat varmasti ensimmäisten päivitysten joukossa. Myös parempi pohja on tarkoitus lisätä. Kameranäkymä on ollut kehitteillä jo ennen tämän projektin alkua, mutta siihen liittyvä teknologia on vielä kehitteillä.

Kaiken kaikkiaan projekti onnistui hyvin. Mevea sai testausvalmiin pohjan, jota se voi testata eri simulaattoreilla, ja äänet toimivat hyvin myös virtuaalitodellisuusympäristössä.

6 Yhteenveto

Insinöörityössä kehitettiin parempi versio jo olemassa olevan kaivinkonesimulaattorin äänimaailmasta ja rakennettiin kokonaan uusi käyttöliittymä, joiden molempien tarkoituksena oli parantaa immersion tunnetta simulaattoria käytettäessä. Näiden päivityksien oli tarkoitus tuoda Mevean omaan simulaattoriin tarvittua nykyaikaista ulkonäköä.

Projektia oli aloitettu jo syksyllä 2022 Metropolian innovaatioprojektina. Insinöörityöprojektissa aloitettiin uuden HMI-pohjan tekeminen, ja tänä aikana tehtiin myös suurin suunnittelutyö. Vaikka kaikkia tämän osan tavoitteita ei saatu suoritettua, lähinnä tiukan aikataulun ja yhden visuaalisen mallinnuksen osaavan henkilön puuttumisen vuoksi, tämän projektin aikana saatiin vietyä loppuun pohjan halutut ominaisuudet ja lisättyä halutut mittarit. Pohjan ilme jäi kuitenkin vanhanaikaiseksi, eikä siihen saatu tarvittavia uusia mittareita.

Äänet onnistuivat kuitenkin hyvin ja saivat hyvän vastaanoton yrityksen sisällä. Uudet taustaäänet olivat hyvä lisä ja toimivat hyvin simulaattorin ympäristössä. Suunnittelu onnistui molempien projektien osalta hyvin, ja yhteistyö oli sujuvaa kaikkien osapuolien kesken. Projekti oli erittäin opettavainen ja mielenkiintoinen kokemus tieto- ja viestintätekniikkaa opiskelevalle, simulaattoreista kiinnostuneelle opiskelijalle. Insinöörityöstä saadut monipuoliset kokemukset opettavat, että pelisovelluksen opinnot mahdollistavat pelialan osaajalle laajan valikoiman työmahdollisuuksia eri aloilta.

Lähteet

1 McMahan, Alison; Wolf, Mark J. P. & Perron, Bernard (toim.). 2003. The Video Game Theory Reader. London: Routledge.

2 Our Story. Verkkoaineisto. Mevea Oy. <https://mevea.com/about/our-story/>. Luettu 8.3.2023.

3 What is HMI? 2018. Verkkoaineisto. Inductive Automation. <https://www.inductiveautomation.com/resources/article/what-is-hmi>. Päivitetty 10.8.2018. Luettu 8.3.2023.

4 Software for Real-Time Simulation. Verkkoaineisto. Mevea Oy. <https://mevea.com/solutions/software/>. Luettu 12.3.2023.

5 Dealessandri, Marie. 2020. What is the best game engine: is Unity right for you? Games Industry. <https://www.gamesindustry.biz/what-is-the-best-game-engine-is-unity-the-right-game-engine-for-you>. Päivitetty 16.1.2020. Luettu 12.3.2023.

6 A tour of the C# language. 2023. Verkkoaineisto. Microsoft Corporation. <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>. Päivitetty 13.2.2023. Luettu 12.3.2023.

7 Takanen, Akseli. 2020. Analysis and Comparison of Unity and FMOD Sound Engines. Bachelor's thesis. Tampere University of Applied Sciences. Theseus-tietokanta.

8 Sekvenssejä. Verkkoaineisto. Peda. <https://peda.net/jklkansalaisopisto/yo/musiikki2/opettajat/riku-heinonen/kitarakurssit/oppimateriaalia2/tekniikkaa/sekvenssej%C3%A4>. Luettu 20.3.2023.

9 FMOD Licensing. Verkkoaineisto. FMOD. <https://www.fmod.com/licensing>. Luettu 15.3.2023.

10 Mevea Oy Yrityshaku. Verkkoaineisto. Kauppalehti Oy. <https://www.kauppalehti.fi/yritykset/yritys/mevea+oy/1979632-8>. Luettu 23.3.2023.

11 What Is a DAW? A Guide to Digital Audio Workstations. 2021. Verkkoaineisto. MasterClass. <https://www.masterclass.com/articles/what-is-a-daw>. Päivitetty 26.8.2021. Luettu 24.3.2023.