LAB University of Applied Sciences

Photon PUN moninpeli Unityssä

Case: VR-HYPO

LAB-ammattikorkeakoulu
Insinööri (AMK) tieto- ja viestintätekniikka, mediatekniikka
2022
Jonni Ala-Vannesluoma

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Ala-Vannesluoma, Jonni	Opinnäytetyö, AMK	2022
	Sivumäärä	
	38	

Työn nimi

Photon PUN moninpeli Unityssä

Case: VR-HYPO

Tutkinto ja koulutusala

Insinööri (AMK), tieto- ja viestintätekniikka

Toimeksiantajan nimi, titteli ja organisaatio (jos opinnäytetyöllä on toimeksiantaja)

CB-Safe projekti, LAB

Tiivistelmä

Työn tavoitteena oli tehdä moninpeli harjoitusalustaa pelastustoimien operatiivisen johtamiseen. Harjoitusalustaan halutiin luoda vähintään kaksi eri onnettomuustilannetta. Lisäksi oli tarkoitus tutkia teknologisia mahdollisuuksia, joilla voidaan parantaa turvallisuutta.

Työssä käytettiin virtuaalitodellisuusteknologiaa, Photon PUN 2 ja Unity-sovellusta harjoitusalustan luomiseen. Näiden pohjalta rakennettiin moninpelisovellus, jossa on mahdollista simuloida ja visualisoida onnettomuustilanteita. Onnettomuustilanteiden rakentamisessa hyödynnettiin turvallisuusalan ammattilaisen tarjoamaa taustatietoa.

Harjoitusalustassa onnistuttiin osoittamaan teknologian hyödyllisyys pelastustoimien operatiivisessa johtamisessa. Työn esittelyssä todettiin, että pelastustoimien operatiivinen johtaminen voisi hyödyntää uusia työkaluja ja teknologiaa turvallisuutta.

Asiasanat

operatiivinen johtaminen, virtuaalitodellisuus, simulointi

Abstract

Author(s)	Type of Publication	Published
Ala-Vannesluoma, Jonni	Thesis, UAS	2022
	Number of Pages	
	38	

Title of Publication

Photon PUN multiplayer with Unity

Case: VR-HYPO

Degree and field of study

Bachelor of Engineering, Information and communication technologies

Name, title and organisation of the client (if the thesis work is commissioned by another party)

CB-Safe project, LAB

Abstract

The goal of the work was to make multiplayer training platform for operational management of rescue operations. Objective was to create at least two different accident situations in the training platform. In addition, the intention was to investigate technological possibilities that can help improving safety.

The work used virtual reality technology, Photon PUN 2 and the Unity application to create the training platform. Based on these a multiplayer application was built, where it is possible to simulate and visualize accident situations. In the construction of accident situations background information provided by a safety professional was used.

In the training platform was possible to demonstrate the usefulness of technology in the operational management of rescue operations. In the presentation of the work was stated that the operative management of rescue operations could make use of new tools and technology for safety.

Keywords

operational management of rescue operations, virtual reality technology, simulate

Sisällys

Termit

Agnostinen

Tietotekniikassa viittaus asiaan, joka on yleistetty niin, että kyseinen asia on yhteen toimiva eri järjestelmien välillä.

API

Application programming interface eli ohjelmointirajapinta.

Determistinen

Viitaus järjestelmään, jossa tietty alkutila tai -ehto tuottaa aina samat tulokset.

ECS

Entity Component System -sanaa käytetään kuvailemaan pelin maailman objektikokonaisuuksia.

Mestari

Master eli määritettyyn palvelimeen luodun moninpelihuoneen haltija. Ei ole palvelimen isäntä.

Prefab

Unity-sovelluksen peliobjektikokonaisuudet, joita luodaan pelin aikana.

RPC

Remote Procedure Call eli etäkäsittelykutsu, jonka avulla toteutetaan koodissa tehty metodi etäkäyttäjillä.

Verkko-objekti

Moninpelin aikana käytettävä peliobjekti, joka on määritetty toimimaan ja synkronisoitumaan verkon kautta.

Viestintäkerros

Käsittelee yhteydet, viestien reitityksen etälaitteiden välillä, ja reitityksen pilven ja laitteiden välillä.

1 Johdanto

Tämä opinnäytetyö on tehty osana Cross Border Safety- projektia, missä tavoitteena on tutkia kykyä visualisoida ja tuoda pelastustoimintaan liittyvää informaatiota esille harjoitusalustassa. Sen pohjalta olisi mahdollista keskustella ja harjoitella riippumatta siitä, ovatko harjoittelijat etänä vai paikan päällä. Lisäksi tavoitteissa tutkitaan mahdollisuudet hyödyntää teknologiaa ja sovelluksia pelastusoperatiivisen johtamisen harjoittelussa, ennakoinnissa ja yleisen turvallisuuden parantamisessa. Lähtökohtaisesti projektissa keskitytään rajaturvallisuuteen, mutta yksi projektin pyrkimyksistä on modulaarinen hyödyntäminen muihin tarpeisiin.

Opinnäytetyön projekti luotiin Unity-ohjelmalla, jossa käytetään Photon PUN 2- integraatiota moninpelin luomiseen ja SteamVR-integraatiota VR-käyttöympäristön tekemiseen. Unity valittiin projektin sovelluspohjaksi kehittäjätiimin yleisen osaamisen ja yhteisöpohjan takia, mikä mahdollistaa erilaisten ratkaisujen löytämisen. Photon PUN 2- integraatio valittiin moninpelialustaksi sen lähestymistavan ja ominaisuuksien vuoksi. SteamVR-integraatio valittiin tätä projektia varten, koska siinä on suuri määrä yhteensopivuuksia erilaisten VR-laitteiden kanssa ja sen tarjoamien Unityn työkalujen takia.

Projekti on LAB ammattikorkeakoulun opiskelijoiden tekemä projekti. Se toteutettiin kahden vuoden aikana pienellä ryhmällä. Projekti tehtiin yhteistyössä LAB:in, Eteläkarjalan Pelastuslaitoksen, KYMPEN ja EU:n kanssa.

2 Moninpelit ja vaihtoehtoja niiden integroimiseen

2.1 Moninpeli

Moninpeli on peli, jota voi pelata useamman henkilön kanssa paikan päällä tai etänä. Video peleissä moninpeli käsittää pelaamisen samalla alustalla tai konsolilla, kuten Playstation (Wikipedia).

Pelaaminen paikan päällä useamman henkilön kanssa tapahtuu jaettujen välineiden kanssa. Video peleissä pelaajat yleensä käyttävät samaan alustaan kytkettyjä ohjaimia, jotka mahdollistavat yhtäaikaisen syötteen peliin. Modernit moninpelit hyödyntävät paljon internet-yhteyttä, joka vähentää tai poistaa tarpeen pelaajilta olla läsnä samassa ympäristössä.

Moninpelit mahdollistavat jaetun kokemuksen pelistä, mikä luo suuremman viihteen arvon. Tämä lisää pelaajien kautta saatavaa palautetta peliin, kuten vastustajien muuttuminen älykkäämmäksi tai yhteistyön tekeminen ongelmien ratkaisemiseksi (Freepubg 2021). Moninpeliympäristöä on mahdollista hyödyntää monessa eri tapauksessa, mikä luo jaetun kokemuksen aiheesta riippumatta. Aiheiden teeman ei tarvitse keskittyä pelilliseen merkitykseen. Jaettu ympäristö voi olla vahvasti vain oppimistilanne, jossa mahdollistetaan informaation jakaminen ja oppiminen interaktiivisella tavalla.

2.2 Unityn moninpeli-integraatio

Unitylle on omat vaihtoehdot moninpelialustojen luomiseen. Alkuperäinen ratkaisu monipelien luomiseksi on Unity UNET, jonka avulla pystyy synkronisoimaan peliobjektit, scriptit ja niiden muuttujat pelaajien välillä. Unity Unet on kuitenkin vanhentunut ja korvattu Unity Netcode-ratkaisulla (Unity3D 2022 a). Unityn moninpeliratkaisun pohjalta on luotu kolmannen osapuolen vaihtoehtoja moninpelien luomiseen, kuten Photon PUN ja Mirror.

Unity Netcode on open-source korkean tason verkkokirjasto, ja sitä saa käyttää ilmaiseksi Unity-projekteissa. Hyötyinä on myös Netcode-ratkaisun käyttämisen yhteensopivuus ympäristönsä kanssa ja se kehittyminen versiopäivitysten mukana. Netcodessa voi synkronisoida peliobjekti- ja maailmandataa verkkoistunnon aikana useammalle pelaajalle kerralla. Ratkaisussa on mukana työkaluja, jotka poistavat tarpeen keskittyä kehittää matalan tason protokollia ja verkkokehyksiä. Sen sijaan nämä työkalut yksinkertaistavat tätä kaikkea ja mahdollistavat keskittymisen projektin rakentamiseen. (Unity3D 2022 b.)

2.3 Mirror

Mirror on moninpeliratkaisu, joka tukee Unity LTS eli Long Term Support versioita (Mirror 2022). Tämä on ilmainen open-source- integraatio pienille indie- tai MMO-peleille, jonka voi ottaa käyttöön ja ladata Unity Storen kautta. Mirror perustuu Unityn oman moninpeliratkaisun pohjalle. Mirror on moninpeliratkaisuna suosittu ja toimiva. Sen avulla on tehty useita erilaisia pelejä, kuten VR-ympäristön moninpeli Population One ja taktinen korttipeli A Glimpse of Luna esimerkiksi. (Mirror 2021.)

Mirror on rakennettu alemman tason kuljetuksen reaaliaikaisen viestintäkerroksen päälle ja hoitaa monia yleisiä tehtäviä, joita moninpeleissä tarvitaan. Sen avulla yksi osallistujista voi olla asiakas ja palvelin samanaikaisesti, joten erillistä palvelinprosessia ei tarvita. Mirrorissa on erilaisia toimintoja, jotka auttavat moninpeliluonnissa. Näihin kuuluu viestien käsittelijät, yleiskäyttöinen korkean suorituskyvyn serialisointi, hajautettu objektin hallinta, tilan synkronointi ja verkkoluokat. (Mirror 2021.)

2.4 SmartFoxServer2X Multiplayer

SmartFoxServer2X on moninpeli-integraatio, joka toimii erilaisissa sovellusympäristöissä. Näihin kuuluu Unity, HTML5, iOS, Android, Java, UniversalWindows Platform, Adobe Flash/Flex/Air ja C++ (SmartFoxServer a). Tämän integraation voi ladata ja ottaa käyttöön Unity Asset Storen kautta, mutta sen omilta kotisivuilta löytyy myös vaihtoehdot sen lataamiseen. Tästä moninpeliratkaisusta on mahdollista ladata ilmainen yhteisöversio, jonka avulla pystyy testaamaan ja luomaan moninpeliprojekteja. Tällä integraatiolla on luotu esimerkiksi hirviökeräilypeli TemTem ja mobiilipeli Shadow Fight 3. (SmartFoxServer b.)

SmartFoxServer-alustassa on erilaisia työkaluja, joiden avulla on mahdollista ideoida ja tehdä erilaisia moninpelejä. Integraation synkronisointi perustuu asiakas/palvelin API:n, jonka avulla voi määrittää virtuaalimaailman arkkitehtuurin vyöhykkeitä, huoneita ja huoneryhmiä, lähettää kutsuja, hallita kaveriluetteloita, luoda mukautettuja käyttöoikeusprofiileja ja valvoa turvallisuusnäkökohtia. (SmartFoxServer b.)

2.5 Photon Engine

Photon Engine on moninpeli-integraatio, jossa on useampia vaihtoehtoja Unity-ohjelmistoon liittyen. Tätä on kuitenkin mahdollista käyttää myös muissa sovellusympäristöissä, kuten Unreal Engine ja kaikki vaihtoehdot perustuvat Photon Cloud-pilvipalveluun. Photon Enginellä tehtyjä pelejä ovat esimerkiksi Osiris: New Dawn ja Nickelodeon Kart Racers 2. (Photon Engine a.)

Photon Engine integraatiot Unity-ohjelmistoon liittyen ovat Photon PUN, Photon PUN 2, Photon Bolt, Photon Fusion, Photon Quantum, Photon Voice ja Photon Chat. Näistä integraatioista uusia päivityksiä saavat vielä Photon Fusion ja Photon Quantum. Photon Voice ja Chat ovat moninpelikommunikointiin liittyvät integraatiot. Kaikki näistä on suunniteltu helpottamaan moninpeli suunnittelua ja yksinkertaistaa työkaluja komponentteihin ja rakennepalikoihin. (Photon Engine a.)

3 Photon Engine

3.1 Photon Realtime

Photon Engine pohjautuu Photon Cloud-pilvipalveluun, jonka päälle on rakennettu Photon Realtime ja tämän päälle itse moninpeliin liittyvät rakenteet. Näihin rakenteisiin kuuluu esimerkiksi Photon Fusion, Quantum. Photon Realtime ratkaisua voi käyttää erilaisissa sovellusympäristöissä, kuten Unreal Engine ja Unity. Realtimen pohjalta tehdyt integraatiot kuitenkin ovat kohdennettu erityisesti Unity-sovelluksen käyttöön. (Photon Engine b.)

Realtime on Photon Enginen pohja moninpeleille, mikä ratkaisee ongelmia peli-istuntojen luomisessa ja kommunikoinnissa skaalautuvalla lähestymistavalla. Kuva 1 osoittaa, kuinka Photon Engine integraatiot rakentuvat ja kaikki lähtökohtaisesti perustuu Photon Cloud-palveluun. Integraatiot pohjautuvat Photon Realtime API:n, jonka pohjalta on jalostettu esimerkiksi Unitya varten Photon Quantum ja Photon Bolt. (Photon Engine b.)

Kuva 1. Photon Engine rakenne (Photon Engine a)

Tarkemmin Photon Realtime API on asiakaspuolen ohjelmisto, jota pelit ja sovellukset käyttävät suoraan. Sen käyttäjät muodostavat yhteyden omistettujen palvelimien sarjaan, jotka jakautuvat nimipalvelimeen, peli-istunnon luomiseen ja pelaamiseen. Realtime tarjoaa asiakkailleen maailmanlaajuisen isännöinnin Photon Cloud kautta, joka on täysin ylläpidetty palvelu. (Photon Engine b.)

3.2 Integraatiovaihtoehdot

3.2.1 Photon PUN 2

PUN 2 eli Photon Unity Networking 2 on Photon Enginen integraatio Unity-ympäristöön. Photon PUN 2:sta on mahdollista ottaa käyttöön ilmainen versio rajoitetulla pelaajamäärällä, jonka avulla on mahdollista harjoitella ja luoda moninpeliprojekti Unity-sovelluksessa. Tähän integraatioon ei tule enää uusia julkaisuja tai ominaisuuksia, mutta se on LTS-muodossa ja käytettävissä. Tämä integraatio on korvannut aiemman Photon PUN-version, mutta pitää sisällään vielä mahdollisuuksia käyttää joitain aiempia komponentteja ja niiden määrityksiä. (Photon Engine c.)

PUN on kopio alkuperäisestä Unityn omasta moninpeliratkaisusta ja se hyödyntää Photon pilvipalvelua. PUN:iin kuuluu erilaisia rakennuspalikoita, joiden avulla moninpeliympäristö Unityssä rakennetaan. Näihin rakennuspalikoihin kuuluu peliobjektien tilojen serialisointi ja etäkäsittelykutsut eli RPC. Lisäksi PUN antaa kehittäjälle suoran ja kokonaisen kontrollin lähetettävästä ja vastaanotettavasta informaatiosta lisättynä sen monilähetyshuonereleviestintämallin kanssa. (Photon Engine c.)

Photon PUN toimii verkkoympäristössä määritetyssä palvelimessa, jonne käyttäjät yhdistävät itsensä ja luovat huoneen tai liittyvät huoneeseen. Huoneille on määritetty mestarit, joilla on enemmän oikeuksia vaikuttaa pelin aikaiseen verkkodataan. Mestari ei ole tässä tapauksessa isäntä tai serveri ja poistuessa huoneesta toinen pelaaja ottaa tämän roolin. (Photon Engine c.)

3.2.2 Photon BOLT

BOLT on myös Unity-ympäristöön tehty vaihtoehtoinen Photonin integraatio, josta on kaksi eri mallia. Ilmainen ja pro-versio, joista ensimmäinen hoitaa kaikki yhteydet Photon pilven kautta. Jälkimmäinen mahdollistaa suorat yhteydet ilman Photon pilveä, omistetut palvelimet ja muokatut asetukset. Kuten PUN, myös BOLT ei saa enää uusia julkaisuja tai ominaisuuksia. (Photon Engine c.)

BOLT on korkeamman tason API, joka mahdollistaa verkkoon soveltuvan pelitilan määrittämisen tietorakenteiden kautta ja liittää nämä resurssit prefab-peliobjekteihin. Nämä peliobjektit hyödyntävät näitä resursseja takaisinkutsuilla, laukaistavilla tapahtumilla ja käskyillä. (Photon Engine c.)

Photon Bolt määrittää verkkoympäristössä palvelimeksi käyttäjän, joka vastaa palvelimen päällä olemisesta. Kyseisen käyttäjän lopettaessa palvelin sammuu ja toiset pelaajat menettävät näin ollen yhteyden palvelimeen. (Photon Engine c.)

3.2.3 Photon Fusion

Fusion on myös Unity-ympäristöön tehty Photonin integraatio, joka tukee kahta erilaista verkko topologiaa ja lisäksi yksinpeli muotoa ilman internet yhteyttä. Fusion toimii, joko jaetussa tai isännöidyssä tilassa. Näiden suurimpana erona on, kenellä on oikeudet muuttaa verkko-objekteja eli muokata näihin liittyvää synkronisoitavaa dataa pelaajien välillä. (Photon Engine d.)

Isännöidyssä tilassa serverillä on täysi ja yksinoikeutettu vaikutus kaikkiin objekteihin ilman poikkeuksia. Client eli vieraskäyttäjät, jotka ovat yhteydessä isännöityyn palveluun, voivat vain muokata verkko-objekteja lähettämällä syötettä serverille tai pyytämällä muutoksen käyttämällä etämetodin kutsua eli RPC:tä. Vieraskäyttäjän päivitys verkkodataan tarkistetaan isännän kautta. Verkkodatan eroa löytyessä, päivittää isäntä vieraskäyttäjän datan oman datan mukaisesti. (Photon Engine d.)

Jaetussa muodossa verkko-objektien oikeudet on jaettu kaikkien vieraskäyttäjien kanssa. Erityisesti käyttäjän luomiin objekteihin on oikeudet, mutta näiden objektien oikeudetkin pystyy toiset käyttäjät antamaan tai ottamaan. Jaettu tila on hyvin samanlainen PUN:in kanssa, mutta täydempi ja nopeampi ominaisuuksiltaan. (Photon Engine d.)

Photon Fusion on kehitetty korvaamaan aiemmat kaksi integraatiota eli Photon PUN ja Photon BOLT. Kaikki näiden integraatioiden tukemat arkkitehtuurit löytyvät myös Fusionista. Vaikka Photon PUN ja Photon BOLT molemmat ovat valmiita integraatioita, ei näiden arkkitehtuureita voi optimoida enempää. (Photon Engine d.)

3.2.4 Photon Quantum

Quantum on determistinen Entity Component System eli ECS-kehys Unityn verkkomoninpeleihin, mikä perustuu ennustamiseen ja palauttamiseen. Tämä sopii parhaiten latenssiherkille verkkopeleille. Quantumissa on mahdollista kirjoittaa koodia, joka erottaa Quantumin simulaatiologiikan näkymästä ja Unityn esittelystä. Samalla Quantum huolehtii verkon toteutuksen erityispiirteistä, joihin kuuluu sisäinen ennakointi ja palautus, siirtokerros ja pelien agnostinen palvelinlogiikka. (Photon Engine e.)

Quantumissa determistinen järjestelmä eroaa normaalista pelaajien syötöstä ja sen tiedon päivityksestä paikalliseen simulaatioon, missä kaikki tämä tieto kerätään jokaiselta pelaajalta ja päivitetään sen jälkeen. Sen sijaan Quantumissa pelaajilla on vapaus paikallisesti viedä simulaatiota eteenpäin käyttäen syöte-ennakointia ja palautusjärjestelmää, joka palauttaa pelin tilat ja uudelleen simuloi virhearvioinnit. Lisäksi peliagnostisen arvovaltaista palvelinkomponenttia hyödynnetään ylläpitämään syöte latenssia ja kellon synkronisointia. Tämän takia käyttäjien ei tarvitse odottaa hitainta palauttamaan tai toteamaan simulaatiota edetäkseen. (Photon Engine e.)

3.2.5 Photon Voice ja Photon Chat

Voice ja Chat ovat Photon Enginen integraatioita, joiden avulla voi liittää äänikommunikoinnin tai tekstikommunikoinnin peliin tai sovellukseen. Voice on mahdollista yhdistää itsenäisenä osana peliä tai pariksi toiseen Photon integraatioon. Photon PUN:iin ja Photon Fusioniin kuitenkin on valmiiksi liitettynä Photon Voice ominaisuudet. (Photon Engine f; Photon Engine g.)

Photon Voice ja Chat molemmat vaativat oman määritetyn palvelin avaimen, joka määritetään projektiin. Joissain integraatioissa kuitenkin erillinen avain ei ole tarpeen, sillä integraatio saattaa itsessään sisältää yhteensopivuuden Photon Voice tai Photon Chat ominaisuuden käyttöön. (Photon Engine f; Photon Engine g.)

4 Photon Network PUN 2

4.1 Kuvaus

Photon PUN 2 on Photon Engine- integraatio, joka toimii kaikissa Unity-ohjelman kanssa yhteensopivissa ympäristöissä. Photon Pun- integraatiosta löytyy tällä hetkellä kaksi eri versiota, joista Photon Pun 2 on uudempi. Photon Pun lisenssistä löytyy ilmainen versio ja maksullinen versio, joista maksullinen tarjoaa kaupallisen käytön ja suuremman pelaajamäärän. Photon Pun 2 on siirtynyt Long term support-muotoon eli LTS, joten uusia ominaisuuksia ei enää tule tähän. Photon PUN 2:een on mahdollista liittää käyttöön sekä Photon Voice ja Photon Chat, jotka mahdollistavat kommunikointityökalujen luonnin moninpeliin. (Photon Engine h.)

Tätä integraatiota käyttäessä on otettava huomioon, että se on suunniteltu internet pohjaisille moninpeliprojekteille ja hyödyntää kaikissa ominaisuuksissa Photon-pilvipalvelua. Tämän takia Photon PUN:in avulla ei voida luoda moninpelejä, jotka eivät hyödynnä internetyhteyttä. Tapa millä PUN 2 toimii, on pelaajien yhdistäminen määrättyyn palvelimeen, johon luodaan aula ja huoneet. (Photon Engine c.)

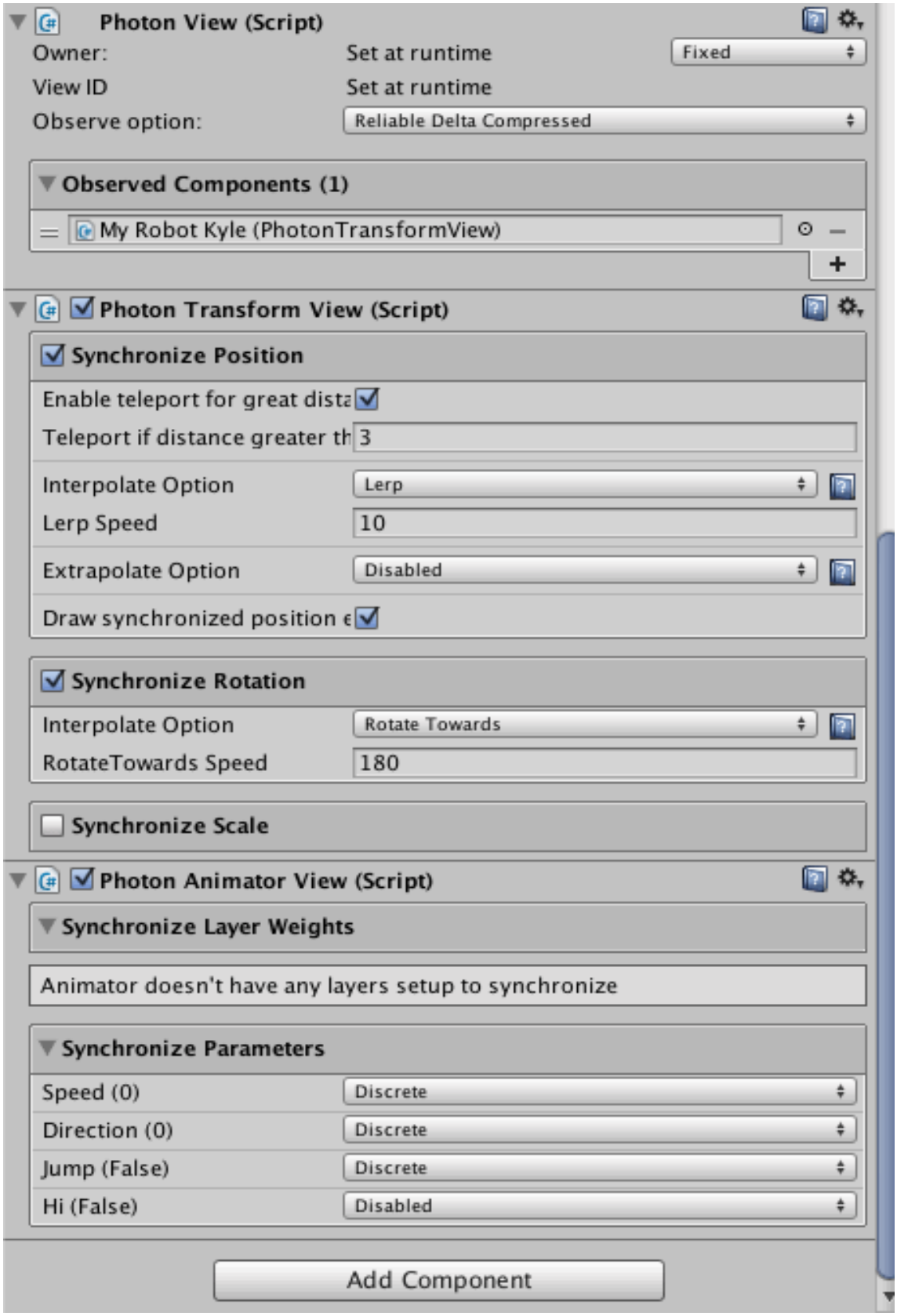
Määritetyllä palvelimella ei ole isäntää, mutta liityttävään moninpelihuoneeseen määritetään mestari. Tämä mestari omaa lisättyjä oikeuksia ja ominaisuuksia verkko-objekteihin ja dataan. Huoneessa on aina yksi mestari, joka korvataan aiemman poistuessa. Kun huoneessa ei ole enää pelaajia, tuhotaan tämä huone. (Photon Engine c.)

4.2 Valmiit komponentit

Photon PUN 2 integraation mukana tulee tähän luotuja valmiita peliobjekteihin liittyviä komponentteja. Näiden komponenttien kautta on yksinkertaisempaa synkronisoida verkko-objektien yleisimpiä käytettäviä komponentteja. Tarpeiden mukaan täytyy luoda komponentteja, jotka käsittelevät samoja tai muita toimintoja projektin tavoitteiden mukaisesti. (Photon Engine j.)

Photon-komponenteista tärkein on PhotonView, joka tunnistaa objektin verkossa ja määrittää sovellukselle objektin synkronisoinnin etäkäyttäjille. Melkein kaikissa peliobjektiin liittyvissä verkon välisessä toiminnassa tarvitaan tämä komponentti. PhotonView-komponentti merkitsee itsensä numerolla automaattisesti tai käyttäjän määrityksellä. Tämä numero erottelee Photon-palvelimelle, mitä tietoa yksittäinen verkko-objekti lähettää tai ottaa vastaan. PhotonView-komponenttin liitetään myös kaikki saman peliobjektin alaisuudessa

olevat Photon-ominaisuuksia käyttävät komponentit kuvan 2 mukaisesti. Nämä komponentit voivat olla peliobjektin juuressa tai lapsissa kiinni. (Photon Engine j.)



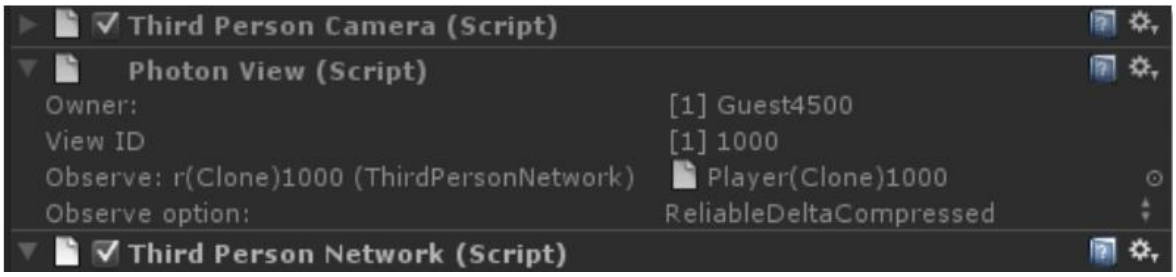
Kuva 2. Peliobjektin PhotonView-asetukset (Photon Engine j)

Muita Photon-komponentteja on erilaisiin tarpeisiin, mutta keskittyvät suurelta osalta synkronisoimaan yleisiä Unity-peliobjektien komponenttien informaatiota, kuten Transform-, Rigidbody-, Animator-komponentteja. Näiden avulla yleiset peliobjektien liikkuminen, koko ja fysiikat on mahdollista synkronisoida pelaajien kesken. Tarvittaessa toisenlaista synkronisointia pitää luoda uusi peliobjektiin liitettävä script-komponentti, johon on liitetty IPunObservable-ominaisuudet. (Photon Engine j.)

4.3 Photon PUN 2 Verkko-objektit

Photon PUN 2 synkronoi verkossa pelaajien välillä verkko-objekteja ja verkkodataa. Nämä verkko-objektit ovat pelissä normaalisti toimivia peliobjekteja, jotka jakavat ja vastaanottavat dataa saman huoneen pelaajien välillä. Verkko-objekteilla on aina määritetyt omistajat, jotka voivat olla huone, mestari tai pelaaja. (Photon Engine i; Photon Engine k.)

Verkko-objektit määritetään kuvan 3 mukaisesti PhotonView-komponentin avulla välittämään dataa Photon palvelimien avulla. Verkko-objektit voivat olla pelin kentässä valmiiksi tai myöhemmin luotuna. Näille objekteille määritetään oletusomistaja, joka usein on pelaaja itse. Määritetty omistaja hallitsee verkko-objektin pelin aikaisen toiminnan dataa, joka synkronisoidaan muille pelaajille. Omistajan poistuessa kaikki tämän pelaajan objektit poistetaan huoneesta. (Photon Engine i; Photon Engine k.)



Kuva 3. PhotonView-komponentti (Photon Engine i)

Verkko-objektit seuraavat synkronoinnissa myös sen hetkistä kenttää. Kun pelissä ladataan uusi kenttä, aiemman kentän aikaiset verkko-objektit poistetaan kentän kanssa. Säilyttääkseen tarvittavat verkko-objektit on näille lisättävä määritykset, jotka säilyttävät kyseiset objektit seuraavaan kenttään. (Photon Engine i; Photon Engine k.)

Verkko-objekteja käyttäessä on otettava huomioon PhotonView-komponenttien raja. Monipelissä tarvitaan paljon eri peliobjekteja, joita luodaan ja synkronisoidaan pelaajille. PUN 2:en raja PhotonView-komponenttia käyttävissä objekteissa on 999, jonka jälkeen sovelluksessa tulee vastaan ongelma peliobjektien luomisessa. Apuna tässä on PhotonView

määrää käsittelemässä automaattinen uudelleenkäyttö. Uudelleenkäytössä vanhat verkko-objektit vapauttavat varatut PhotonView-paikat uusille objekteille poistamisen yhteydessä. (Photon Engine i; Photon Engine k.)

4.4 Photon PUN 2 Callbacks

Callback eli takaisinkutsu-metodit ovat funktioita, jotka toteutuvat tietyn tapahtuman tai työn jälkeen. Nämä auttavat Photon Networkissa luomaan toimintoja tapahtumien toteutuessa kuvan 4 mukaisesti. Näihin tapahtumiin kuuluu yleiset moninpeliin liittyvät verkkotapahtumat, kuten liittyminen ja poistuminen Photon-palvelimille. Myös monet huonekohtaiset ja pelaajakohtaiset tapahtumat ovat osa takaisinkutsuja. (Photon Engine l; Photon Engine m.)

3. Let's add the following Photon callbacks messages and save `GameManager` script

```
#region Photon Messages


public override void OnPhotonPlayerConnected(PhotonPlayer other)
{
    Debug.Log("OnPhotonPlayerConnected() " + other.NickName); // not seen if you're the pl

    if (PhotonNetwork.isMasterClient)
    {
        Debug.Log("OnPhotonPlayerConnected isMasterClient " + PhotonNetwork.isMasterClient

        LoadArena();
    }
}


public override void OnPhotonPlayerDisconnected(PhotonPlayer other)
{
    Debug.Log("OnPhotonPlayerDisconnected() " + other.NickName); // seen when other discon

    if (PhotonNetwork.isMasterClient)
    {
        Debug.Log("OnPhotonPlayerDisonnected isMasterClient " + PhotonNetwork.isMasterClie

        LoadArena();
    }
}


#endregion
```

Kuva 4. PUN Callbacks (Photon Engine p)

Photon PUN 2:en takaisinkutsujen käyttäminen tarvitsee oman koodikirjastonsa MonoBehaviourPunCallbacks-luokan, joka periytetään Unityn script-komponenttissa. Näiden hyödyntäminen helpottaa integraation käyttöä ja useimmat ovat yhteydessä toisiinsa tai lähellä toisiansa tapahtumajärjestyksessä. Yksinkertaisen moninpelin luominen on mahdollista ainoastaan käyttäen takaisinkutsuja, joiden käyttäminen myös osoitetaan Photon PUN 2- integraation mukana tulevissa esimerkeissä. (Photon Engine l; Photon Engine m.)

4.5 Photon PUN 2 Custom Properties

Erilaisia asetuksia pelaajiin ja huoneisiin liittyen on mahdollista määrittää erillisillä Custom Properties- asetuksilla. Tyypillisesti huoneet ja pelaajat pitävät sisällään erilaista tietoa, joka ei ole liitoksissa peliobjekteihin. Tällaista tietoa on esimerkiksi käytössä oleva kenttä ja pelaajan ulkonäkö. (Photon Engine n.)

Asetukset eivät rajoitu ulkonäöllisiin ominaisuuksiin, vaan mahdollistavat erilaisten asetusten tai tietojen välittämisen pelaajille Photon-palvelimien kautta. Näille asetuksille löytyy omat takaisinkutsut, jotka mahdollistavat erilaisten tapahtumien tekemisen riippuen mitä ja miten on muutettu. (Photon Engine n.)

Sekä pelaajakohtaiset ja huonekohtaiset tiedot päivitetään hashtable-muuttujalla, joka määritetään Photon koodikirjaston mukaisesti. Pelaajakohtaiset tiedot voidaan jokaiselle pelaajalle määrittää ainutlaatuiseksi. Tämän kautta on mahdollista erotella pelaajat täysin erilaisiin rooleihin tai toimintoihin. Huonekohtaiset voidaan määrittää vain yhtenä joukkona ja päivitetty joukko korvaa aina edellisen. Tämän takia huonekohtaisia tietoja on suositeltavaa sallia vain yhden pelaajan päivittää, minkä avulla vältetään ongelmatilanteita. (Photon Engine n.)

4.6 Photon Pun 2 RPC ja Event

Photon PUN-integraatiolla on mahdollista välittää eri käyttäjien ohjelmille tiedoksi, että täytyy toteuttaa tietty funktio. Tähän on luotu Remote Procedure Calls eli RPC-funktio, joka nimensä mukaisesti kutsuu metodin toteutusta etäkäyttäjillä samassa huoneessa (Photon Engine o). Metodi tarvitsee määrityksen [PunRPC], joka merkitsee funktion RPC-metodiksi. Tämä ilmoittaa Photon-asetuksille, että tätä funktiota kutsutaan Photon-palvelimien kautta, joka helpottaa monimutkaisempien toimintojen toteuttamisen pelaajilla. (Photon Engine o.)

Jokainen RPC määrittää automaattisesti PhotonServerSettings-tiedostoon listauksen kyseisestä funktiosta koodin kääntämisen yhteydessä. Tämän listauksen avulla Photon PUN 2 tarkastaa projektiin luodut RPC-funktiot. Listaus ei kuitenkaan automaattisesti poista RPC-funktioita, mikä vaatii listan tyhjentämisen ja funktioiden uudelleen etsimisen. Näitä toimintoja varten on PhotonServerSettings-tiedostossa erilliset käskyt Unity-editorissa. (Photon Engine o.)

RPC on suunniteltu käyttämään samaa PhotonView-määritettyä verkko-objektia kutsuessa, jonka takia on otettava huomioon PhotonView-komponentin numerointi. Tämä numerointi on oltava sama RPC-kutsun yhteydessä, jotta oikea objekti yrittää käyttää tätä funktiota. Lisäksi verkko-objekteissa on oltava komponentti, joka sisältää saman nimisen RPC-funktion. (Photon Engine o.)

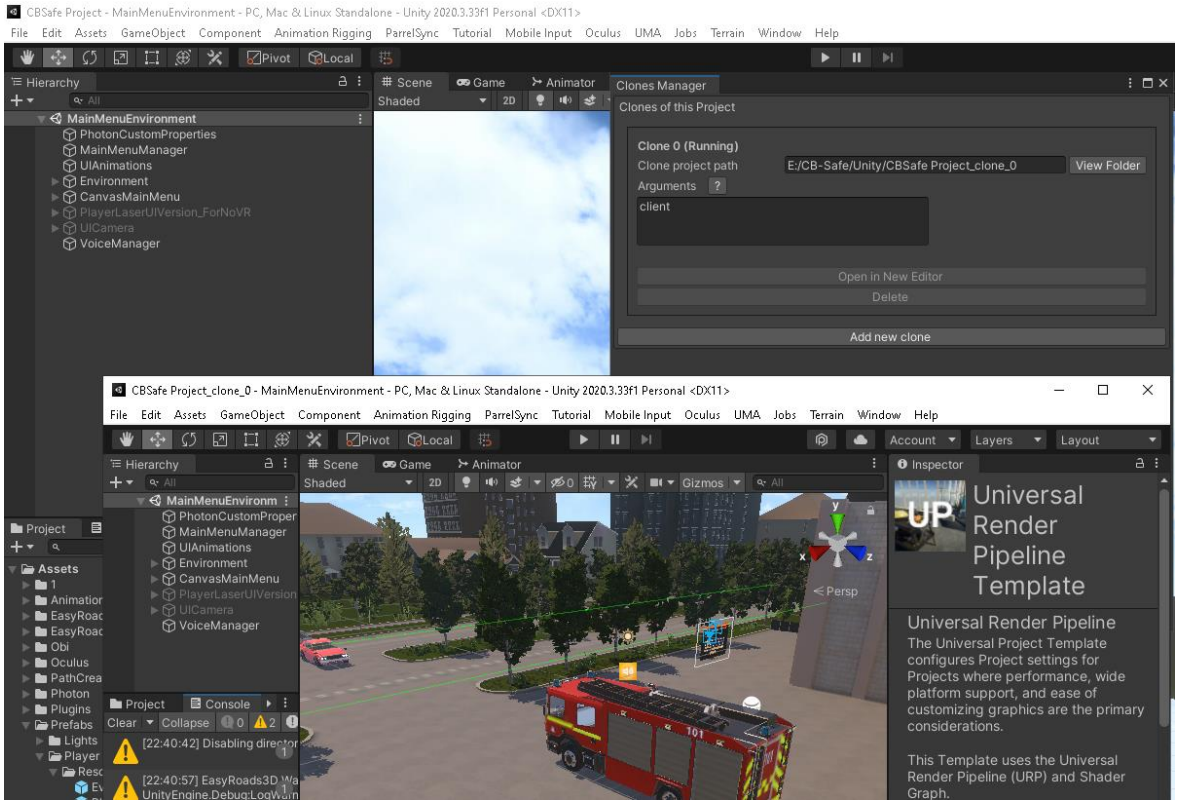
Photon Event eli Photonin mukautettu verkkotapahtuma on samanlainen RPC-funktion kanssa, mutta eroaa tarpeesta olla liitettynä ja kutsuttuna peliobjektin kautta. Nämä tapahtumat erottelevat itsensä bitteinä, mikä mahdollistaa 256 erilaista tapahtumaa. Photon on varannut osan biteistä itselleen sisäänrakennettuihin tapahtumiin, mutta mahdollistaa silti 200 eri tapahtuman määrittämistä. (Photon Engine o.)

Sekä RPC-funktiossa ja Photon tapahtumassa on mahdollista määrittää kutsun aikana vastaanottavat pelaajat kutsun asetuksissa. Tämä mahdollistaa erittelyn erilaisissa verkkotapahtumissa pelaajien välillä, mikä antaa lisää joustavuutta toimintojen määrittelemisessä. (Photon Engine o.)

4.7 Moninpelin testaaminen

Moninpelin testaaminen on välttämätön vaihe, jossa tarkistetaan rakennetun sisällön toimivuus ympäristössä. Testaaminen onnistuu yksinkertaisesti luodun sovelluksen samanaikaisella käyttämisellä editorin kanssa tai useamman laitteiston kanssa, joista jälkimmäinen on suositeltavampaa. Toisin kuin yhden henkilön käytössä, erilaista tietoa välitetään muiden sovelluksiin synkronisoiden pelitilaa tai toteuttaen komentoja. Näiden takia virheenluku eroaa tavallisesta. (Unity3D 2021.)

Sovelluksen luominen jokaisen korjauksen tai muutoksen jälkeen on aikaa vievää, minkä takia Unity-editoriin on tehty ParrelSync-työkalu. ParrelSync monistaa editorin ja jakaa projektin tiedostot tämän editorin kanssa kuvan 5 osoittamalla tavalla. Tämä mahdollistaa useamman kopion sovelluksesta ajamaan testausta yhdellä laitteella samanaikaisesti. (Unity3D 2021.)



Kuva 5. ParrelSync-työkalu

Photon-integraation moninpelin testauksessa seurataan Photon-takaisinkutsujen toimivuus ja niiden toteutus, PhotonView-komponenttien mahdolliset virheet, lähetettyjen tietojen yhteensopivuus tiedon lukemisen kanssa ja tapahtumien kulku suhteessa odotettuun. Testaamisen vertaaminen suunniteltuun toteutukseen helpottaa ongelmien etsimistä tapahtumien aikana. Ongelmien ja virheiden kartoittamiseen auttaa myös Unityn oma debug-komento. Mitä enemmän informaatiota ohjelma ilmoittaa eri vaiheissa, sen helpompi on seurata ongelmakohtia. (Unity3D 2021.)

5 CB-Safe VR-HYPO Simulaatiosovellus

5.1 CB-Safe Projekti

Tämä simulaatio on osa Cross Border Safety eli CB-Safe-projektia, joka koostuu useammasta työpaketista ja simulaatio kuuluu työpaketti kakkoseen. Simulaatio tehtiin Unity-ohjelmalla VR-ympäristöön. Sovellukseen oli tarkoitus luoda kaksi eri skenaariota, missä voi simuloida pelastustehtäviä ja mahdollistaa uusien skenaarioiden lisääminen myöhemmin. Projektin oli tarkoitus samalla osoittaa erilaisia mahdollisuuksia, joita on mahdollista saavuttaa hyödyntäen erilaista teknologiaa ja ohjelmistoa. Teknologialla oli tarkoitus löytää tapoja vähentää tai helpottaa erityyppisiä toimenpiteitä, millä voidaan saavuttaa valmiutta pelastustilanteisiin. Projektissa mukana oli yhteistyössä LAB, Eteläkarjalan Pelastuslaitos, KYMPE ja EU.

Simulaation sovelluksen luontia varten tekijöille jaettiin pohjamateriaalia, joka sisälsi toimintamalleja pelastusoperaatioihin, tietoa ja tiedostoja luotavaan ympäristöön liittyen. Tekijät myös etsivät itse materiaalia ja tutustuivat simulaation tavoiteympäristöön Saimaan kanavan Mälkiän Sulun luona, jossa ottivat valokuvia ja keskustelivat projektin toteutuksesta.

Projekti ajoitettiin vuoden 2021 kesältä vuoden 2022 marraskuun loppuun. Tämän aikavälin aikana projektia testattiin säännöllisesti ja esiteltiin kohderyhmälle, joilta pyydettiin palautetta sovelluksen hyödyllisyydestä ja sen tulevaisuuden näkymistä. Projektin sovellusta on ollut luomassa ryhmä opiskelijoita, jotka kuuluivat LAB ammattikorkeakoulun tieto- ja viestintätekniikan koulutusohjelmaan. Opiskelijat osallistuivat projektiin osa-aikaisesti kesäopiskelijoina ja tekivät projektia pääasiallisesti yhden kesän ajan.

5.2 VR-ympäristö

Simulaation VR-ympäristö toteutettiin SteamVR-integraatiolla, jolla on yhteensopivuus useimpien laitteiden kanssa. Tämän integraation liittäminen projektiin onnistui suoraan Unity Asset-sivustolta, jossa löytyy kyseinen tuote ilmaiseksi ladattavaksi.

Suurella osasta VR-laitteita on omat tarvittavat ajurit, jotka on asennettava käytettävälle tietokoneelle. VR-laiteet eroavat toisistaan tavasta seurata käyttäjän liikkeitä ja annettua informaatiota. Yleisimmät tavat VR-laitteilla on seurata käyttäjän pään ja käsien liikkeitä,

jotka tapahtuvat VR-laseilla ja molempien käsien ohjaimilla. Lisäksi VR-laitteilla on usein paikan ja laitteiden seurantatavoista, joko sisäinen seuranta tai ulkoinen seuranta.

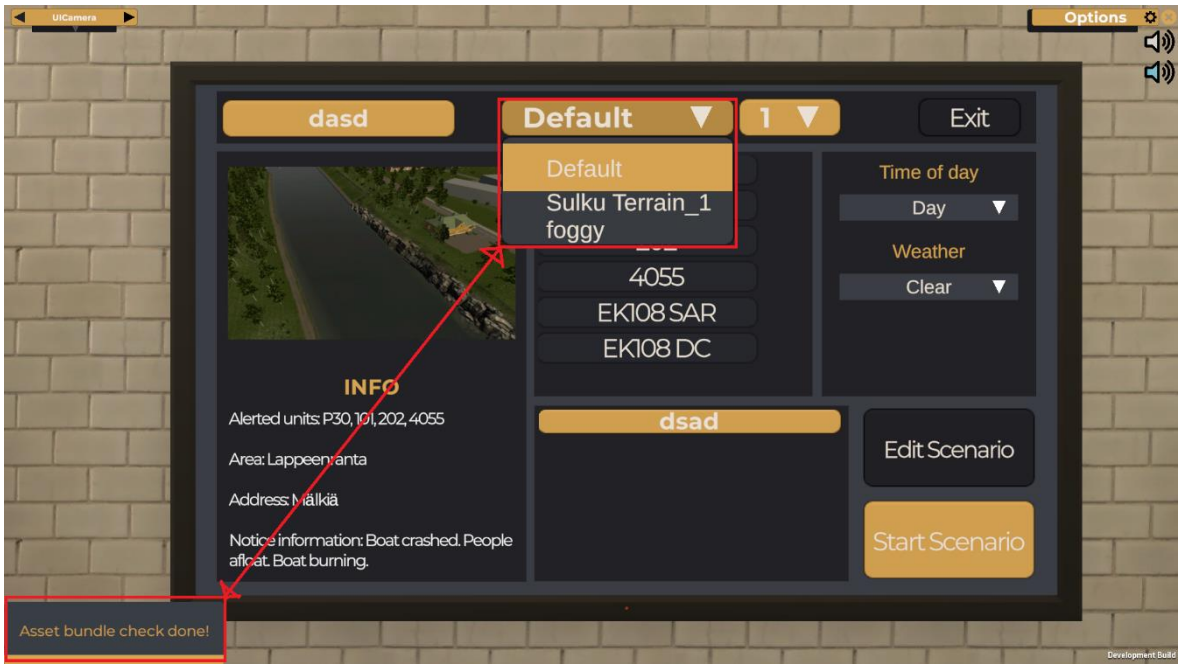
SteamVR-integraation mukana tulee suurin osa kaikista tarvittavista ominaisuuksista, mitä VR-projekti saattaa tarvita. Monet tarvittavat ominaisuudet on kuitenkin tehtävä itse tai ostaa. VR-ympäristön hyödyntäminen eroaa tavallisesta 3D-projektista ja tarvitsee VR-ympäristön käyttämiseen soveltuvan tietokoneen. Lisäksi on otettava huomioon erilaiset grafiikka-asetukset, jotka Unity mahdollistaa, ja näiden pohjalta valita paras vaihtoehto projektin tarpeisiin. Tämän projektin tarpeisiin valittiin Universal Rendering Pipeline eli URP-asetus. Tämä mahdollisti hyvän näköisen ulkoasun ilman liian raskasta kuormaa laitteistolle.

5.3 Modulaarisuus

Tämän projektin yksi tavoitteista oli modulaarisuus eli sovelluksen ominaisuuksien käyttäminen eri tilanteissa tai mahdollisuus laajentaa niitä. Tässä tapauksessa teknologisessa ja erilaisten simulaatiotilanteiden muodossa. Tämä modulaarisuus oli tarkoitus mahdollistaa valmiissa sovelluksessa ilman, että erikseen tarvitsee luoda uusi sovellus lisäyksen jälkeen.

Toteutus lisäyksille tehtiin käyttämällä Unityn omaa tapaa luoda Unity Asset Bundle-tiedostoja. Tämä tiedosto on editorissa luotu merkityistä resursseista tehty kokoonpano, joka luetaan projektin streaming assets-kansiosta. Tällä tavalla voidaan rakentaa projektiin ominaisuus, joka lukee kyseisen kansion tiedostot ja lataa sieltä tarvittavan simulaatioympäristön.

Moninpelissä vaaditaan, että jokaisella pelaajalla on sama Asset Bundle-tiedosto samalla nimellä ja tämän tiedoston lataaminen samassa tilanteessa. Tämän projektin tapauksessa asset bundle-tiedostoja hyödyntävä ominaisuus oli laitettu kuvan 6 osoittamaan kenttään, jossa valitaan huoneen simulaatioasetukset ja kaikki asset bundle-tiedostot tarkastetaan kenttään tullessa. Kaikki simulaation asetukset valitsee huoneen mestari, joka välittää RPC-komennolla tarvittavat tiedot ladattavasta asset bundle-tiedostosta huoneen pelaajille.



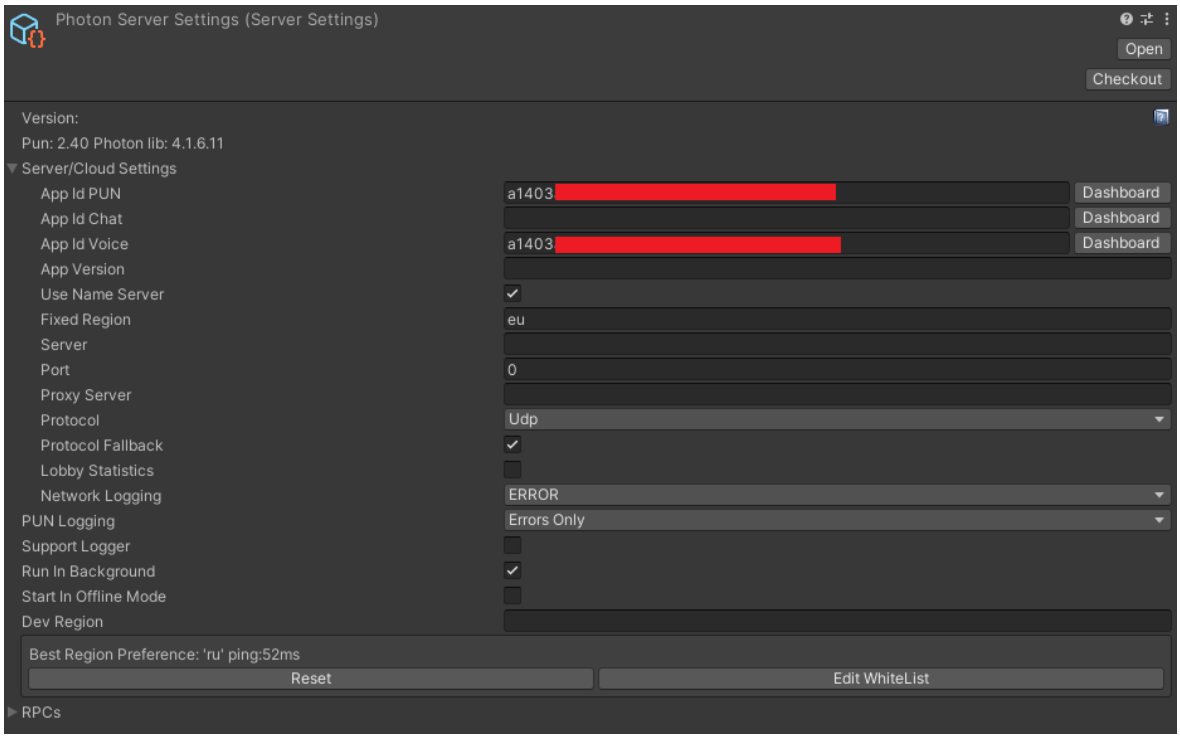
Kuva 6. Unity Asset Bundlen tarkistus

Simulaation alkaessa lähettää huoneen mestari RPC-kutsun huoneen pelaajille, mikä käskee sovellusta lataamaan valitun kentän listasta. Jos tämä kenttä on asset bundlesta tuotu, RPC-kutsu lataa tarvittavat tiedostot ja kentän. Ladattavaan kenttään on määritetty tarvittavat peliobjektit ja rakenteet, jotka mahdollistavat samanlaisen toiminnan alkuperäisen kentän tavalla.

5.4 Photon PUN 2 valmistelu

Photon PUN 2 valittiin tähän projektiin, sillä sen nähtiin täyttävän tarpeet moninpeli ympäristön rakentamisesta haluttuihin tavoitteisiin. Tärkeimpinä tavoitteista nähtiin helppo yhdistäminen samaan peli-istuntoon ja useampien tilanteiden mahdollistaminen eri simulaatioryhmille. Lisäksi puhekommunikointi koettiin vahvasti tarpeellisena simuloidakseen tilanteeseen liittyvää vuoropuhelua, joka saavutettiin Photon Voice ominaisuudella.

Photon PUN 2 vaatii palvelin avaimen, joka määrittää minne Photon pyrkii yhdistämään projektia moninpeli tilanteessa. Tämä avain määrittää samalla myös, minkä tyypin sovellus on kyseessä. Kuva 7 osoittaa, minkälaiset asetukset säädöt vaaditaan projektiin, joka käyttää ääni kommunikointia ja moninpeliä Photon PUN 2 kautta.

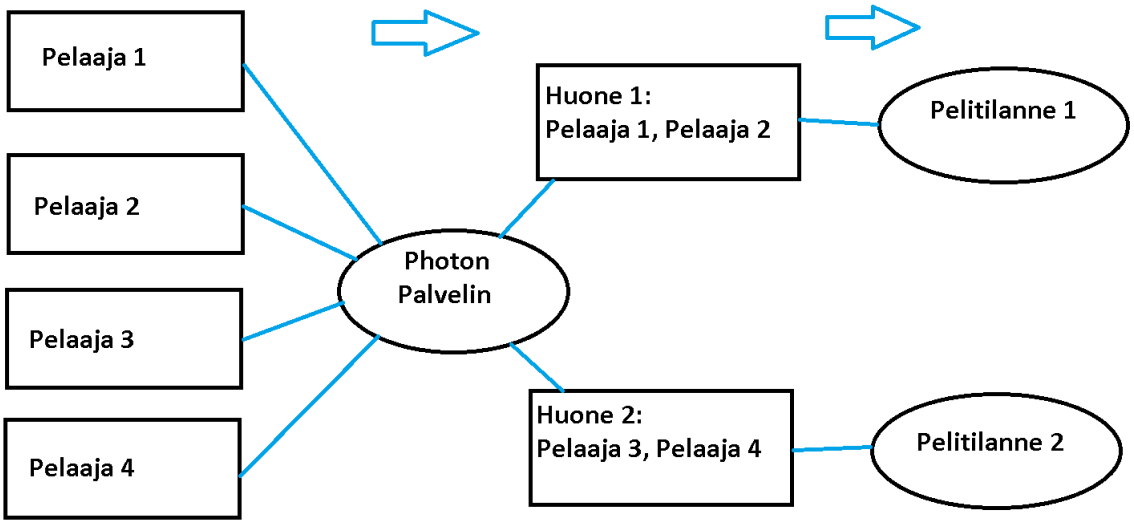


Kuva 7. Photon asetukset

Kuva 7 mukaisesti riippuen integraatiosta on mahdollista käyttää samaa avainta, jonka avulla tuodaan useampia ominaisuuksia projektiin. Avain on mahdollista luoda kirjautumalla Photon Engine sivustoon ja siellä määrittää se tarvitsemaansa integraatioon. Huomioon on otettava, että eri integraatiot omaavat erilaiset lisenssit palveluun liittyen.

5.5 Pelaajien yhdistäminen peliin

Projektin lähtökohtaisen tarpeen perusteella, tapa miten pelaajat yhdistetään simulaatiotilanteeseen ja millainen määrä pelaajia yhdistetään samaan tilanteeseen, parhaimpana vaihtoehtona nähtiin huonekohtainen yhdistäminen. Kyseinen tapa yhdistää pelaajat listattuun huoneeseen, joita voi olla useampia ja eri tilanteissa. Nämä huoneet jakavat samassa huoneessa olevien synkronisoitavan tiedon toistensa kanssa kuvan 8 osoittamalla tavalla.



Kuva 8. Huonekohtainen moninpeli

Tämä mahdollisti projektin suhteen useamman simulaatiotilanteen pitämisen erilaisilla määrityksillä ja opetustilanteilla. Jokaiseen simulaation huoneeseen kuuluu mestari, joka vastaa simulaatioasetuksista. Huoneen mestari voi olla kuka tahansa, mutta sovellus asetettiin valitsemaan simulaation valvoja oletetusti. Jokainen huone koostuu maksimissaan 8 pelaajasta, joista yksi voi olla valvoja. Valvojien määrä on rajattu huoneessa yhteen.

Huonekohtaisesti ja mestarin avulla päätetään myös, kuka on vastuussa simulaatioon liittyvien tietojen synkronoimisesta pelaajille. Simulaation aikana on mahdollista olla paljon erilaisia peliobjekteja yhtäaikaisesti luoden suuremman taakan laiteresursseille huoneen mestarille, jonka takia simulaation valvoja koettiin hyvänä vaihtoehtona olla vastuussa simulaation objekteista.

Jos simulaatio on jo käynnissä, uuden pelaajan liittyminen huoneeseen aiheuttaa ongelmatilanteita. Näiden ongelmatilanteiden välttämiseksi luotiin rajoitus simulaation käynnistyessä, joka estää uusien pelaajien liittymisen kyseisen simulaation huoneeseen. Lisäksi pakotettiin nimien yksilöllisyys, jotta vältetään sekaannus pelaajien ja huoneiden kesken.

5.6 Simulaatio ympäristö

Simulaation ympäristö luotiin projektin tavoitteiden mukaisesti Saimaan kanavan Mälkiän Sulun alueesta, missä simulaatiopelaajan on tarkoitus liikkua ja katsoa ympärilleen VR-laitteiden avulla. Tämän takia kyseinen ympäristö rakennettiin pohjamateriaalissa annetun pistepilvi-tiedoston avulla vastaamaan oikeaa kokoa.

Moninpelitilanteessa pitää pystyä varmistamaan, että pelaajat lataavat saman kentän. Kentän kautta pelaajille tuodaan sama ympäristö ja toiminnot. Photon PUN 2:ssa tämä tapahtuu käyttämällä omassa koodikirjastossa löytyvää komentoa Photonnetwork.LoadLevel(), joka välittää käskyn ladata kenttä saman huoneen pelaajille.

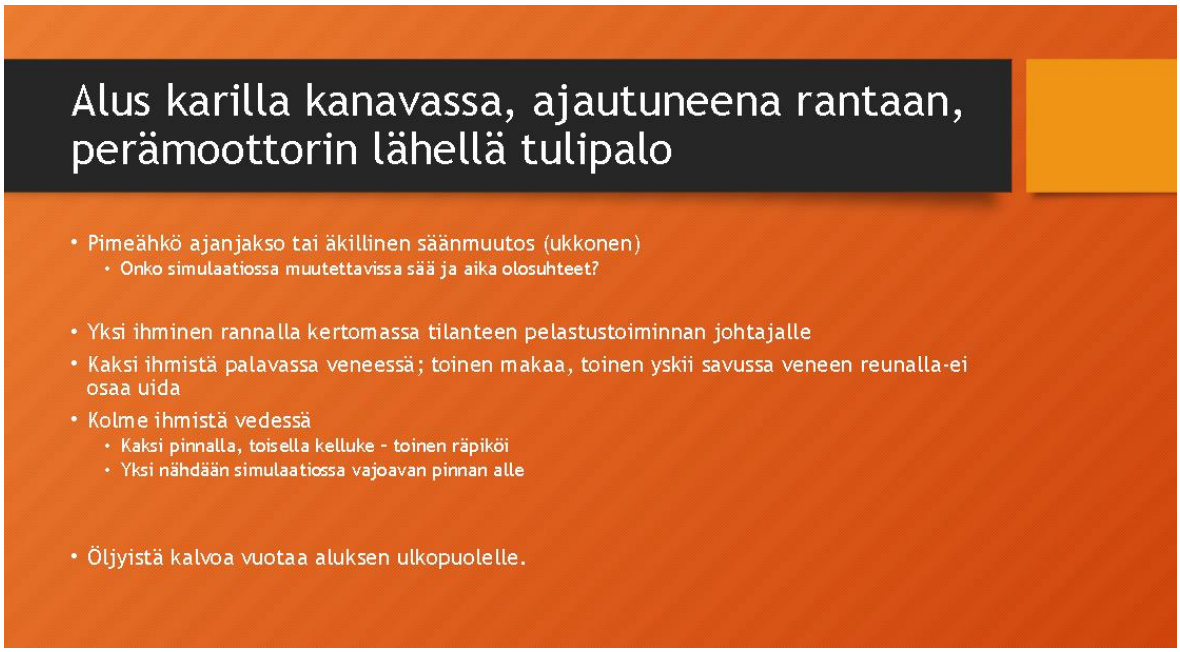
Tämän projektin simulaatio on jaettu kolmeen eri sceneen eli kenttään kuvan 9 mukaisesti. Avauskentän tarkoitus on mahdollistaa pelaajan määrittää erilaisia asetuksia, kuten nimensä simulaatiossa ja luotavan tai liityttävän huoneen nimen. Täältä pelaaja siirtyy itse huoneeseen, jonne samaan simulaatiotilanteeseen osallistuvat saapuvat. Tässä kentässä pelaajat kykenevät kommunikoimaan tai katsomaan toistensa ulkomuotoa ja liikkumista. Päätavoitteena tässä kentässä on valmistella tarvittavat asetukset seuraavaa kenttää varten ja tarkistaa pelaajien valmius. Huoneesta siirrytään simulaatiokenttään, jossa itse ympäristö, tilanne ja yksiköt luovat rakenteen kentälle.

Kuva 9. Simulaation kentät Main Menu, Lobby ja Scenario

5.7 Simulaation tapaukset

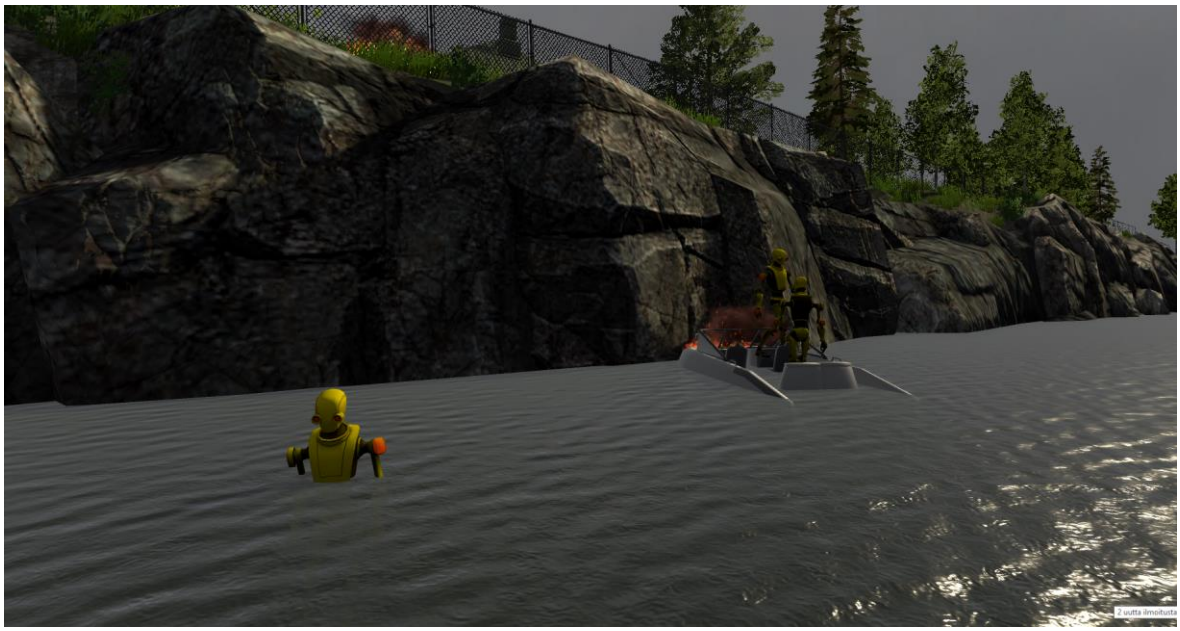
Simulaation tapausten rakentaminen aloitettiin annettujen käsikirjoitusten pohjalta ja tavoitteena oli luoda vähintään kaksi erilaista tapausta simulaatiota varten. Projektin aikana pelastustoimen ammattilainen konsultoi tapauksiin ja pelastustoimenpiteisiin liittyen. Kuvan 10 mukaiselta pohjalta tapausta rakennettiin projektiin, mihin pyydettiin säännöllisesti

palautetta. Tapaukset rakentuivat onnettomuustilanteen ympärille, siihen vaadituille pelastusyksiköille ja mahdollisuudelle pyytää tai kutsua lisäapua. Lisäksi näissä on muita liikkuvia osia, joiden takia näiden tapausten rakentaminen aloitettiin visualisoiden erilaisilla saatavilla olevilla materiaaleilla. Tähän pohjaan ryhdyttiin lisäämään älykkyyttä ja toimintaa.



Kuva 10. Simulaation käsikirjoitus (Toni Salmi 2021)

Toimintaa rakennettiin onnettomuustilanteiden erilaisten osien, kuten tulipalojen, öljyvuotojen ja pelastettavien henkilöiden kautta kuvan 11 osoittamalla tavalla. Näille osille etsittiin ja pyydettiin tietoa. Tiedon pohjalta tarkennettiin osien käsittelyä pelastustoimissa ja niiden mahdollista käyttäytymismallia, jonka pohjalta simulaatiotoimintaa näille rakennettiin.



Kuva 11. Tapauksen rakennetta

Kun osilla on rakennettu riittävästi toimintaa, ryhdyttiin luomaan pelastustoimien tekoälypohjaa. Se käyttäytyy simulaatio tilanteen mukaisesti ja kuvan 12 mukaisesti rakennettiin itsenäiseen toimintaan perustuviin simulaation osiin, kuten yksiköt ja pelastajat. Näitä varten kartoitettiin mahdollisimman paljon pelastustoimintoja ja käyttäytymistä pohjamateriaalin lisäksi. Projektin edistyessä näiden osien ominaisuuksia ja visuaalista rakennetta päivitettiin aina tarpeen mukaan lähemmäksi projektin tavoitteita.

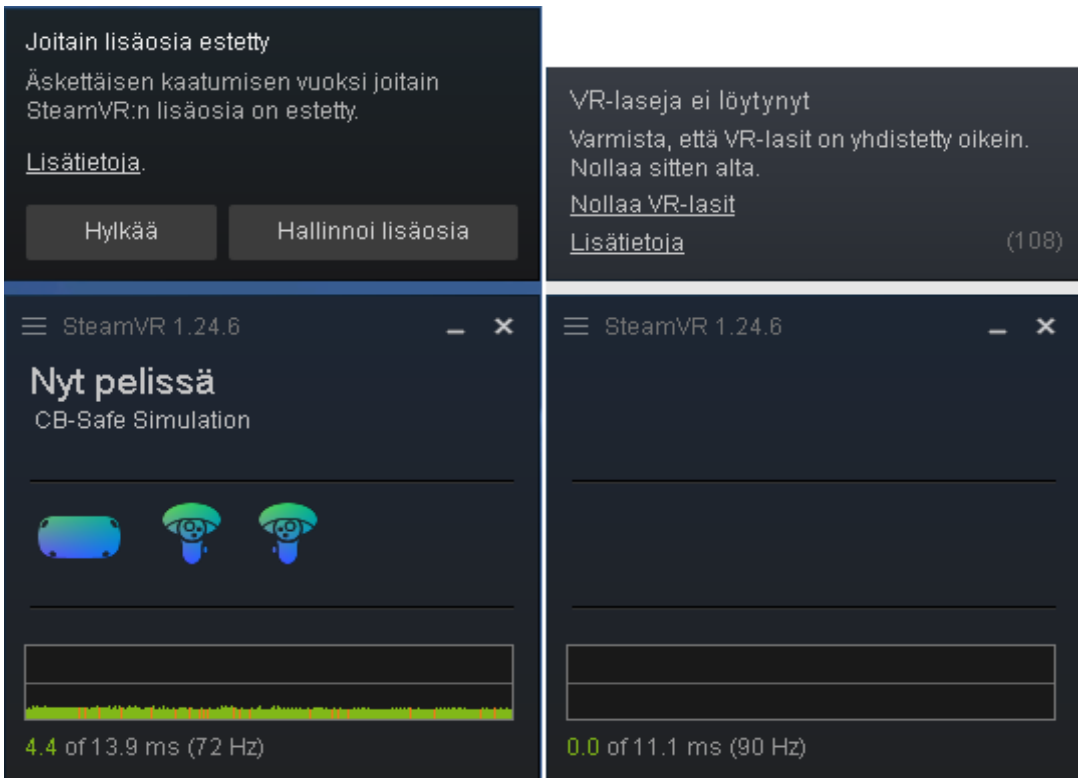
Kuva 12. Tekoälypohjan komponentteja

5.8 Simulaation valvoja

Erilaista palautetta kerätessä projektiin pyydettiin simulaation valvojan ominaisuutta. Tämän ominaisuuden olisi tarkoitus mahdollistaa simulaation ja pelaajien ulkoisen vaikuttajan ja arvioinnin tuomisen sovellukseen. Valvoja kykenee näkemään ja vaikuttamaan suurempaan määrän informaatiota simulaatiotilanteessa. Tämän takia päätettiin, että valvoja ei käytä VR-laitteita. Ominaisuutta varten täytyi erotella VR-pelaaja ja tavallinen pelaaja sovellusta käyttäessä.

Erottelua varten löytyi kaksi vaihtoehtoa. Näistä ensimmäinen oli tehdä Unitylla kaksi eri sovellusta, joissa on eri asetukset ja komponentit. Toisena vaihtoehtona oli tarkistaa VR-laitteiden kytkentä VR-integraation kautta ja tämän mukaan aktivoida tarvittavat komponentit. Koska projektin testaaminen ja sen rakentaminen sovellukseksi vievät aikaa, päätettiin käyttää jälkimmäistä vaihtoehtoa.

Valvojan ja VR-pelaajan erottelu tässä projektissa tehdään heti sovelluksen käynnistyttyä, jolloin projektin käyttämä SteamVR-integraatio käynnistää oman sovelluksen ja ilmoittaa yhdistetyn VR-laitteen kuvan 13 mukaisesti. Tämän tiedon kautta pystyttiin määrittelemään, mikä versio pelaajasta otetaan käyttöön ja mitkä komponentit aktivoidaan. Koska VR-pelaajan ja valvojan ero on lähtökohtaisesti jo erilainen, vaaditaan kaksi erilaista versiota pelaajasta peliobjektina.



Kuva 13. SteamVR-ohjelmiston ilmoitukset yhdistetyistä laitteista

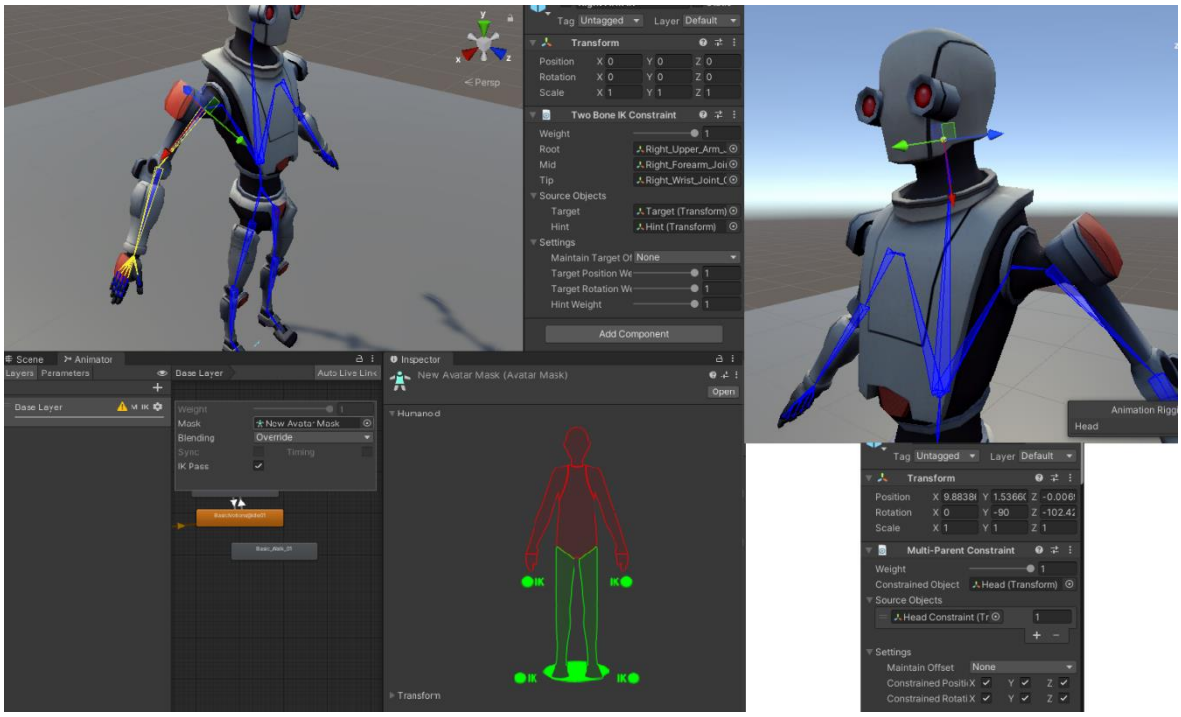
5.9 VR-pelaaja

VR-pelaaja toiminnoiltaan, interaktiivisuudeltaan ja käytöltään eroaa yleisestä pelaajasta huomattavasti VR-laitteiston takia. VR-pelaaja kykenee liikkumaan fyysisesti virtuaaliympäristössä, jossa muutetaan oman kehon liikkeet sovelluksessa käytettäväksi tiedoksi. Tämä mahdollistaa erilaisen käytettävän informaation ja interaktiivisuuden immersion kanssa huomattavasti suuremmaksi. Samalla tavoin eroaa informaatio, jota toiset pelaajat voivat nähdä muiden pelaajien suhteen.

Kuvan 14 mukaisesti VR-pelaaja toimii yhdessä muiden VR-pelaajien kanssa simulaatiossa, jossa pelaajat keskustelevat ja liikkuvat ympäristössä. Yhdessä toimimista varten VR-pelaajille luodaan virtuaalinen keho eli avatar, joka näkyy ja animoidaan muille pelaajille. Tämän avatarin kautta pelaajat myös tietävät etäisyyden toisistaan kommunikoinnin osalta, joka jakaantuu etäisyyteen perustuvaan äänikommunikointiin ja radiokommunikointiin.

Kuva 14. VR-Pelaaja valvojan silmin

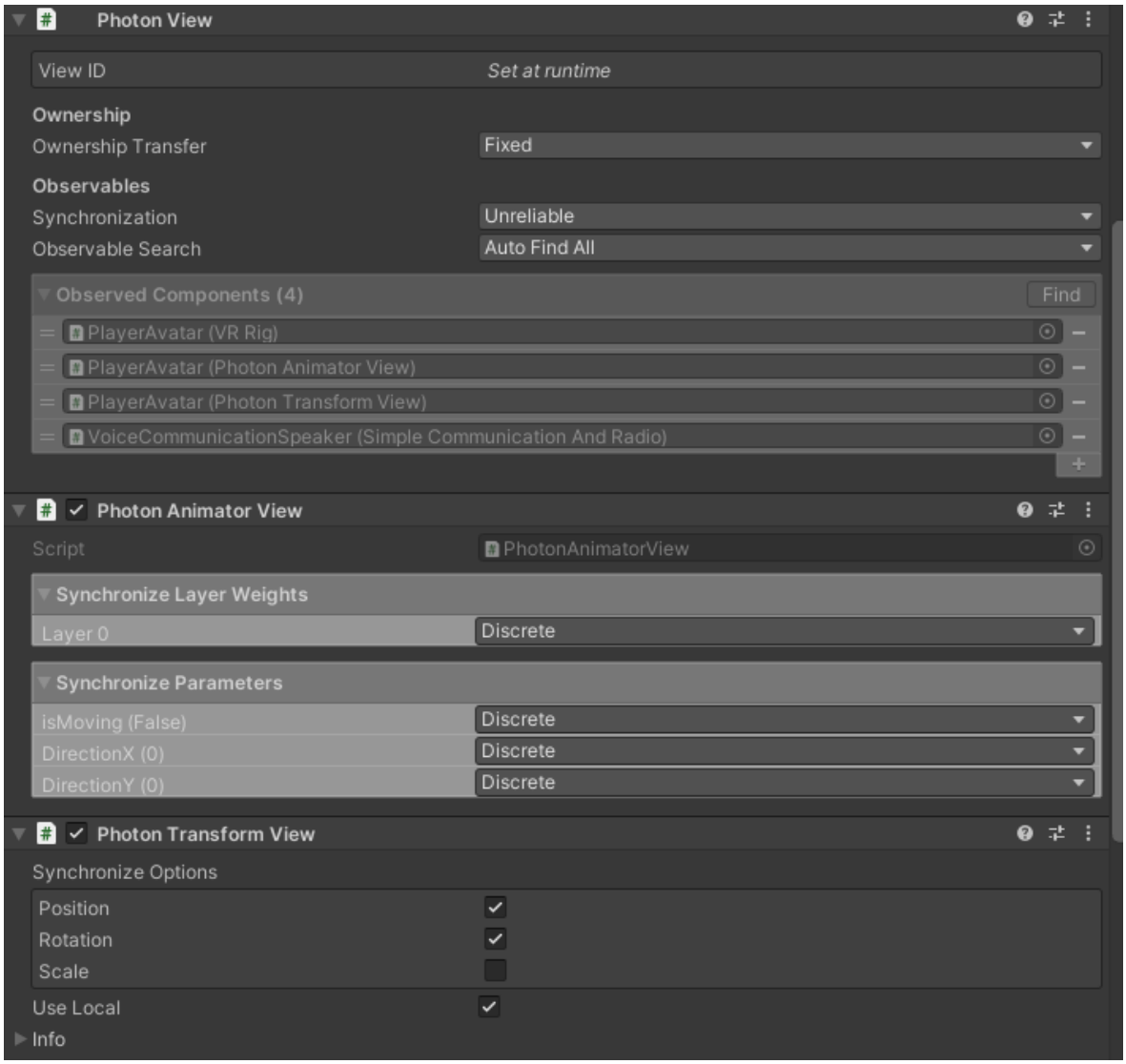
Avatar määrittää animaation luurangon liikkumaan pelaajan omien liikkeiden mukaisesti. Yleisesti VR-laitteisto määrittää pelaajan korkeuden, pään ja käsien liikeradan. Kuvan 15 mukaisesti Unityssä on mahdollista määritellä animation rigging- työkalujen avulla avatarin liikkumista pelaajan VR-laitteiden mukaisesti. Päätä ja käsiä varten kaikille määritettiin Inverse Kinematics- ominaisuudet, jotka auttavat avatarin luita liikkumaan pelaajan liikeinformaatiota seuraten.



Kuva 15. Virtuaalikehon määrittämistä

Koska jaloille ei ole seurantalaitteita, määritettiin jalat liikkumaan animaatioiden avulla. Nämä animaatiot asetettiin havainnoimaan pelaajan avatarin liikkeen 3D-maailmassa. Suurin osa animaatioista ovat koko kehoa animoivia, minkä takia täytyi määrittää uusi animaatiokerros animoimaan tarvitut alueet.

Avatarin määritysten lisäksi tarvitaan asetukset tiedolle, jota synkronoidaan moninpelin aikana muille pelaajille. Avatarin liikkeet voidaan synkronisoida muokatuilla määrityksillä tai Photon Animator View-komponentilla, joka synkronisoi peliobjektin Animator-komponentin tiedot muille pelaajille. Kuvan 16 mukaisesti virtuaalikehon Photon-komponenttien asetuksiin tehdään tarvittavat synkronointiasetukset. Lisäksi moninpeliä varten määritetään avatarin luomistilanne, joka tässä projektissa päätettiin määrittää pelaajien liittyessä huoneeseen ja liittyvän pelaajan omistukseen.

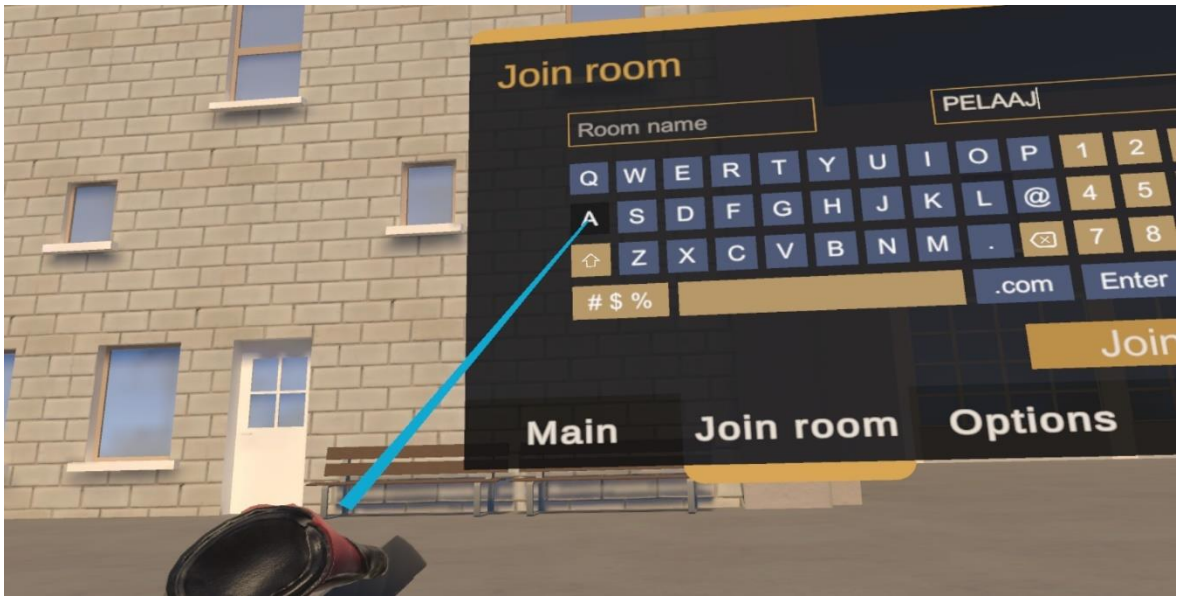


Kuva 16. Avatar komponenttien määrityksiä

5.10 Käyttöliittymä

Käyttöliittymä eli User Interface jakaantuu kahteen eri tyyppiin riippuen, onko sovellus käytössä VR-laitteen kanssa vai ilman. VR-pelaaja ei voi käyttää normaalia sovelluksen ikkunaan liitettyä UI:ta kameran renderöintitavan eron takia. VR-kamera renderöi sovelluksessa vasemman ja oikean silmän yhtäaikaisesti muodostaen syvyysnäkymän, jonka takia VR-laitteen käyttäjälle normaali UI ei näkyisi oikein.

VR-pelaajaa varten käyttöliittymä on oltava sijoitettuna ympäristöön pelattavan maailman koordinaatteja käyttäen, joka on muotoa Vector3. Kuvan 17 mukaisesti UI:sta muodostuu kyseisellä tavalla osa 3D-ympäristöä ja renderöityy VR-kameran molempiin silmiin. Lisäksi ympäristöön sijoitettua käyttöliittymää on mahdollista käyttää myös ilman VR-laitetta.



Kuva 17. 3D-ympäristön UI

Projektin käyttöliittymissä jaetaan tietoa eri asioihin pelaajien välillä. Niiden takia kyseisiin käyttöliittymiin suunniteltiin tarvittavat UI-objektit ja niille informaatio, joka jaetaan Photon PUN 2:en välityksellä. Kuvan 18 UI:ssa jaetaan pelaajille useampaa muutettavaa tietoa, kuten huoneessa olevat pelaajat, pelaajien valitut yksiköt, valittu kenttä ja kentän informaatioteksti. Photon PUN 2:en kautta voidaan jakaa tietoa tietyntyyppisissä muuttujien muodossa, kuten string-muuttuja. Kuvan 18 UI:ssa jaetaan tekstipohjainen tieto string-muuttujien avulla ja tarvittava määrä UI-objekteja yksiköiden valintaan luodaan toteuttamalla objektin luontikäsky pelaajille, jotka myös tuhotaan samalla tavalla tarvittaessa.

Kuva 18. Lobby-kentän UI

Photon PUN 2 omaa erilaisiin verkkotapahtumiin liittyen työkaluja, joiden avulla voidaan tapahtuman ajankohdalla määrittää Unity-sovellus tekemään asioita. Projektin käyttöliittymissä haetaan näiden tapahtumien avulla tietoa liittyen huoneisiin, pelaajiin, yhdistettyyn alueeseen, sovelluksen aikaiset pelaajat ja yhteiset muuttujat helpottaen haluttujen ominaisuuksien tekoa.

5.11 Pelaajien kommunikointi

Simulaatiossa pelaajat pyrkivät toimimaan ja keskustelemaan simulaation tilanteesta ja ympäristöstä pelastustoimien johtamisen näkökulmasta. Lisäksi projektin tavoitteena on mahdollistaa sovelluksen käyttäminen etätoiminnassa, jonka takia moninpelin aikainen äänikommunikointi koettiin tärkeäksi ominaisuudeksi. Tämä toteutettiin projektissa Photon PUN 2:en mukana tulevalla Photon Voicen toiminnallisuudella.

Projektin alkuperäisessä suunnitelmassa äänikommunikointiin pyrittiin luomaan kyky käyttää puhekommunikointia ja radiokommunikointia, joka mahdollistaa yksittäisen keskustelun eri ryhmien kanssa ja on visualisoitu pelaajalle kuvassa 19. Projektin tilanne muuttui kehityksen aikana, minkä takia radiokommunikoinnin tavoitetta yksinkertaistettiin yhteen ryhmään. Lisäksi äänikommunikoinnissa otettiin huomioon simulaation valvojan läsnäolo, joka kykenee keskustelemaan pelaajien kanssa etäisyydestä riippumatta.

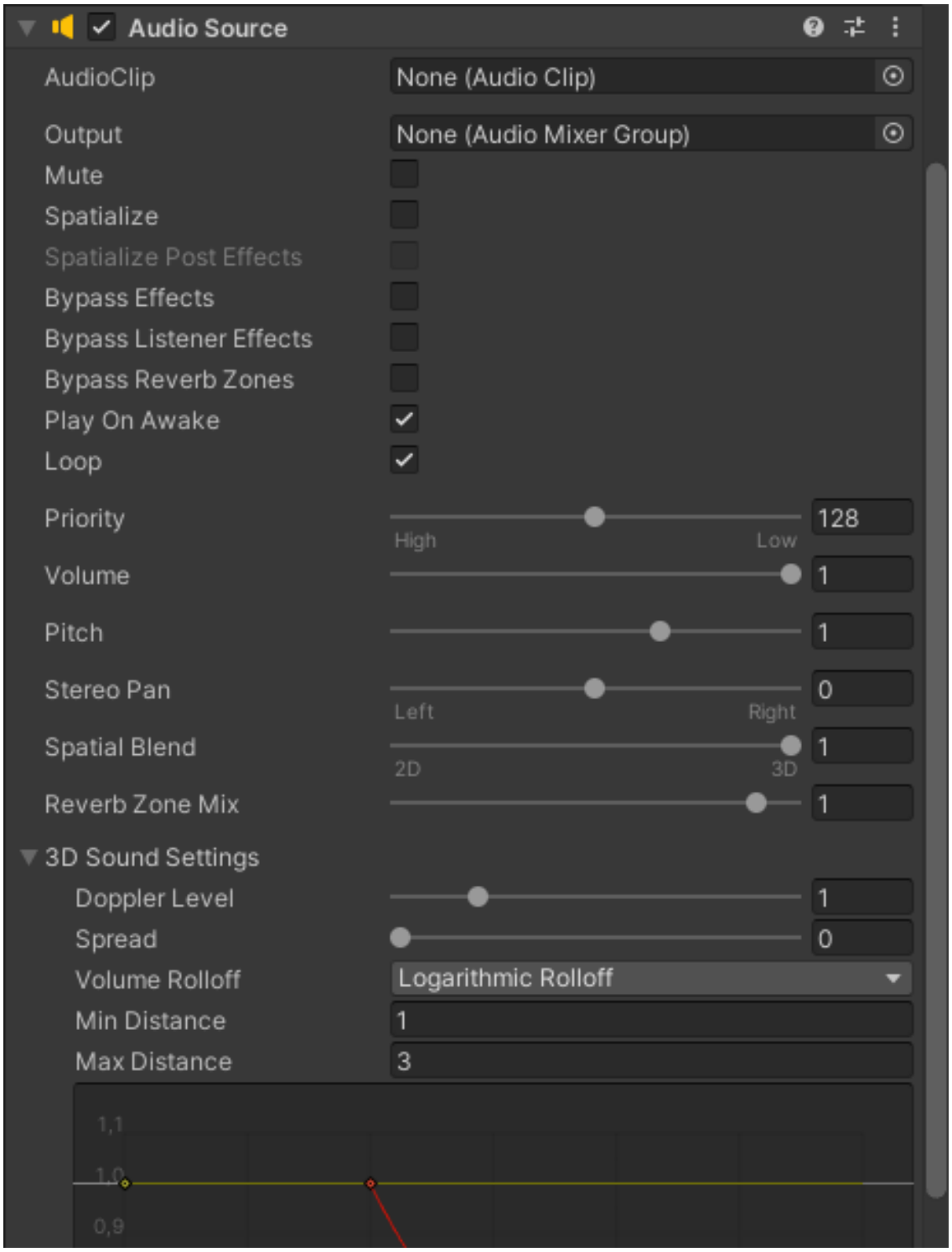
Kuva 19. VR-pelaajan radio

Äänikommmunikoinnin luonti aloitettiin VR-pelaajalla, jolle luotiin ensimmäisenä etäisyyteen perustuva puhekommunikointi. Photonilla on esimerkkiprojektit tämän luomiselle, jonka pohjalta lähdettiin rakentamaan äänikommunikointia. Photonin esimerkki käyttää projektissaan hyödyksi collider-komponentteja, joiden avulla tunnistetaan pelaajan olevan

puhe-etäisyydellä. Tämän suhteen erottiin tavasta tarkastella etäisyyttä hyödyntämällä Unityn audio source- komponenttia, johon on mahdollista asettaa äänen kuuluvuuden etäisyys. Photon Voice käyttää äänikommunikointitavassaan Photonin Recorder- ja Speaker- komponentteja liitettynä Unityn Audio Source- komponenttiin, minkä takia on mahdollista muuttaa nämä tiedot suoraan Audio Source- komponentin kautta.

Photon Voice käyttää Photon Voice Network- komponenttia, joka määrittelee äänikommunikointi asetukset ja yhteyden Photon palvelimien välillä. Photon Voicen komponentit ovat vahvasti yhteydessä Photon Voice Network- komponenttiin, joita voi olla vain yksi kerralla olemassa pelin aikana. Tämän takia äänilähteet, jotka tarvitsevat Recorder-komponenttia, tarvitsevat Photon Voice View-komponentin. Tämä komponentti mahdollistaa ääniasetusten liittämisen erilliseen peliobjektiin, jossa on Photon Speaker- komponentti. Näiden toimintojen pohjalta VR-pelaajan avatariin liitettiin Photon Speaker- komponentti, joka mahdollistaa etäisyyteen perustuvan puhekommunikoinnin.

Radiokommunikointi projektissa mahdollistaa pelaajien äänikommunikoinnin etäisyydestä riippumatta samassa huoneessa olevien pelaajien kanssa. Photonin esimerkit toteuttavat tämän tyypin ominaisuuden Interest Group- ryhmien määritysten avulla, mitkä erottelevat pelaajien kommunikointiin liittyvät kuuluvuusryhmät. Projektissa toteutuksessa erottiin käyttämällä Unityn Audio Source- komponenttia kuvassa 20. Audio Source- komponentin spatial-asetusta muuttamalla voidaan määrittää, onko ääni 2D vai 3D ympäristössä. Erona näissä on, että ensimmäinen kuuluu etäisyydestä huolimatta ja jälkimmäinen huomioi etäisyyden. Tämä asetus ratkaisussa synkronisoidaan pelaajien välillä, kun VR-pelaaja painaa määritettyä nappia luoden radiokommunikointiominaisuuden.



Kuva 20. Audio Source-komponentti

VR-pelaajan äänikommunikoinnin pohjalta tehtiin simulaation valvojalle äänikommunikointi. Valvojalla ominaisuus eroaa ainoastaan tarpeesta kuulla kaikki pelaajat jatkuvasti, valikoivasti kommunikoida kaikille pelaajille ja puhua silminnäkijän roolissa. Tämä erotel-

tiin projektissa kahdella eri kommunikointinapilla pelaajia ja silminnäkijää varten. Silminnäkijä toteutettiin samalla periaatteella, miten VR-pelaajan radiokommunikointi 3D-ympäristössä toimi.

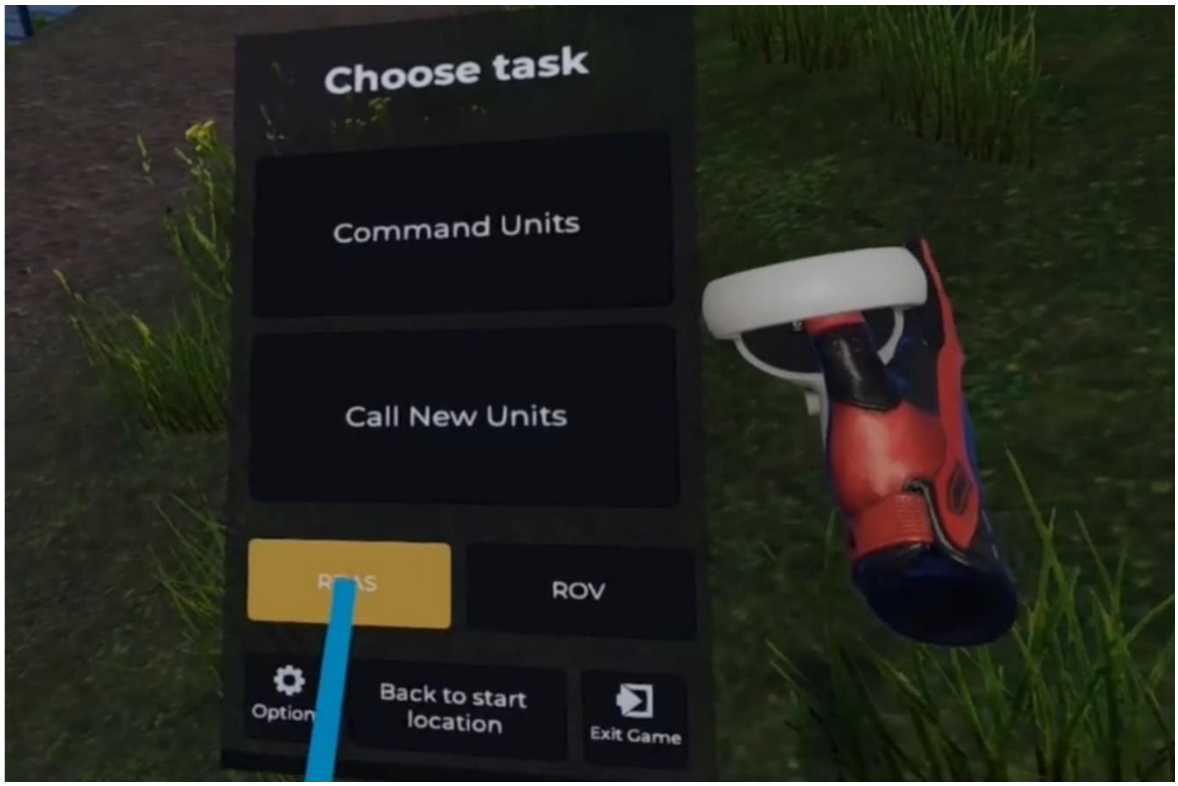
Valvojan normaalia äänikommunikointia varten tehtiin yleinen äänilähde, joka luodaan saapuessa moninpelihuoneeseen. Tämä kaiutin välittää ääntä 2D ympäristössä kuuluen etäisyydestä huolimatta. Valvojan puhe ei kuulu jatkuvasti pelaajille, jonka takia käytössä oleva recorder-komponentin äänen nauhoitus täytyy asettaa käyttöön vain nappia painaessa. Kuullakseen VR-pelaajat jatkuvasti, valvojaa varten määritettiin lisäasetus VR-pelaajien äänilähteiden suhteen. Tämä asetus tarkistaa pelaajan roolin valvojana ja jättää äänen 2D-asentoon.

5.12 Pelaajien toiminnot

Simulaatiossa on tapahtumia, joita pelaajat tekevät vaikuttaakseen simulaation kulkuun ja tilanteisiin. Näitä varten pelaajat tarvitsevat pelillisiä toimintoja, joihin tapahtumat liitetään. Näillä toiminnoilla tavoiteltiin pelastusyksiköiden johtamista ja simulaatiokentän muokkaamista. Äänikommunikoinnin kanssa näistä muodostuisi visuaalisella palautteella harjoitusalusta operatiivisen johtamisen pelastustoimiin.

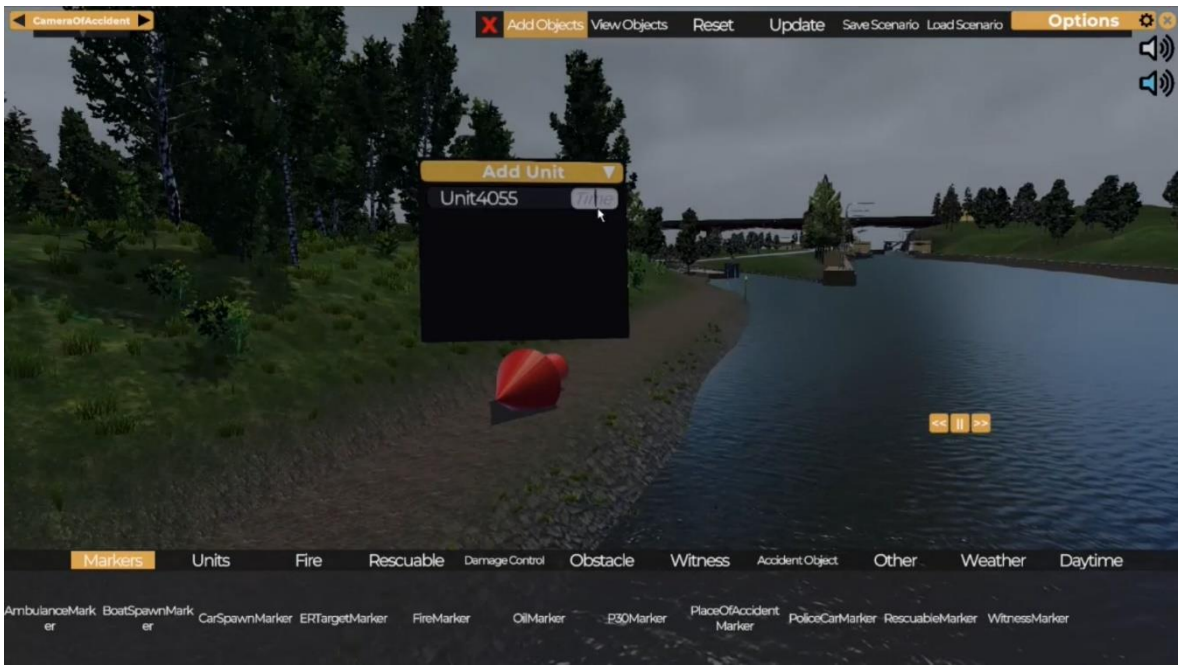
Pelilliset toiminnot ovat erilaiset erityyppisillä käyttöliittymillä VR-pelaajalla ja valvojalla. Nämä erotettiin toisistaan huomioiden ero VR-laitteissa ja peliobjektien aktiivisuudella. VR-pelaaja ei pysty näkemään tavallista ruutuun perustuvaa UI:ta, jota hyödynnettiin valvojan toimintoja luodessa. VR-pelaajan käyttöliittymä on 3D-ympäristössä ja näkyisi valvojallekin, mutta tämä ratkaistiin VR-pelaajaobjektin olemassaololla. Tähän objektiin on liitetty VR-pelaajan käyttöliittymät.

VR-pelaajalla on kuvan 21 mukaisesti toimintoja pelastusyksiköiden tekoälyyn liittyen ja nämä jaettiin kahteen eri tyyppiin. Nämä ovat simulaatiossa pelastustoimien johtajan ja yksikön ryhmänjohtajan toiminnot. Toiminnot lähettävät moninpelihuoneen mestarille, joka ylläpitää simulaation tekoälyllisiä tapahtumia ja objekteja, Photonin tapahtumia ilmoittaen halutusta toiminnasta tekoälylle.



Kuva 21. VR-pelaajan toimintoja

Simulaation valvojalla toiminnot liittyvät simulaation tilanteeseen ja objekteihin vaikuttamisella, jotka rakennettiin kuvan 22 mukaisella tavalla. Toimintojen ja informaation määrän takia täytyi nämä sovittaa käyttöliittymään käytännöllisesti. Näiden avulla valvoja on mahdollista luoda simulaatiota varten uusia objekteja tai vaikuttaa näihin.



Kuva 22. Valvojan toimintoja

5.13 Moninpelikokonaisuus

Kun kaikki tarvittavat osiot pelissä rakentuvat projektin ympärille, täytyy ne sovittaa mahdollisimman hyvin kokonaistoimintaan. Tässä projektissa testausta toimintojen suhteen tehtiin säännöllisesti, mutta itse moninpelitestaaminen jäi projektin loppupuolelle. Tämä vaikeutti toimintojen toteamista halutulle aikavälille ja suurensi työtaakkaa kiireen takia. Palautetta ja testausta kuitenkin kerättiin riittävästi kehittämään kokonaisuutta paremmaksi.

Projektin visuaalinen osuus onnistuttiin kokoamaan ensimmäisen kesän aikana, mutta siinäkin myöhemmin todettiin tarvetta optimoida kevyemmäksi ja tehokkaammaksi. Moninpelitilanteessa tuli vastaan huomattavasti enemmän erilaisia ongelmatilanteita, joiden korjaaminen vaati tarkkaa optimointia ja objektien tarkistamista. Optimoinnin kautta visuaalinen rakenne VR-ympäristössä tehtiin mahdollisimman toimivaksi.

VR-ympäristön toteutus onnistui ensimmäisen kesän aikana, mutta vaati huomattavan määrän toiminnallista hienosäätöä koko projektin ajan. Simulaatio muuttui tarpeesta olla täysin VR-ympäristössä testausten ja palautteiden aikana, jolloin pyydettiin ominaisuutta simulaation valvojalle. Tämä vaati VR-ympäristön käyttämistä ja kehittämistä dynaamisesti eri tilanteissa, joissa vaadittiin uudelleen sovittamista projektiin.

Projektin toiminnallisuus rakentui jatkuvasti uusien ominaisuuksien kanssa, jotka pystyttiin liittämään käyttöliittymään tarpeen tullen. Haasteena oli sovittaa toiminnallisuus muuttuviin tarpeisiin projektin aikana, kuten valvojan toiminta pyydettiin puolen vuoden jälkeen palautteissa.

Projektin tekoälyosuutta kehitettiin projektin alusta saakka, mutta eri painoarvoilla ja tavoilla. Kehityksen painoarvo kasvoi vasta viimeisen vuoden aikana, jolloin myös tekoälyn toteutustapa muuttui. Simulaation synkronisointi pelaajille täytyi testata ja todeta useampaan kertaan muutoksen yhteydessä. Lopulta päädyttiin projektissa käyttämään Photon tapahtumia ja huoneen mestaria tekoälyn synkronisointia varten.

Moninpeliosuutta tehtiin projektin alusta asti, mutta vaiheittain. Toteutus aloitettiin yksinkertaisimmilla moninpelin ominaisuuksilla, kuten huoneen luominen ja siihen liittyminen. Tämän jälkeen kehitystä jatkettiin pelaajien välisellä synkronisoinnilla, jotta kommunikointi pelaajilla onnistuisi. Aiempien moninpelin toteutuksen vaiheiden jälkeen oli mahdollista kehittää simulaation aikaisille toiminnoille testausympäristöä, jossa moninpelin ja sen toimintojen pystyttiin testaamaan.

Kaikkien osioiden rakentumisen ja sovittamisen jälkeen kaikkia edellä mainittuja ominaisuuksia testattiin jatkuvasti. Testauksella etsittiin mahdollisia ongelmakohtia ja virheitä, joiden kautta paranneltiin ja korjattiin kokonaisuutta. Tässä vaiheessa simulaation toiminnallisuus ja ominaisuudet paranivat huomattavasti. Samalla saatiin parempi kuva simulaation käyttötarkoituksesta ja konseptin mahdollisuuksista. Kun sovellus läheni projektin tavoiteltua valmiutta, esiteltiin ja testattiin simulaatiota kohdehenkilöiden kanssa.

6 Yhteenveto ja pohdinta

Projektin laajuuden takia ei ollut mahdollista saada täysin valmista sovellusta aikaiseksi. Kuitenkin onnistuttiin osoittamaan teknologian ja sovelluksien mahdollisuudet harjoitusalustalla. Tavoitteiden osalta projekti onnistui osoittamaan kaikki pyydetyt asiat ja enemmän, mitä alkuun oli pyydetty.

Projektia varten olisi tarvittu enemmän aikaa ja tekijöitä. Lisäksi pelastustoimien konsultointia todettiin tarvittavan lisää sovelluksen toimintojen sovitusta varten. Haasteeksi syntyi myös tarve projektiin sidotuille kehittäjille, jotka olisivat mukana kehityksessä projektin loppuun asti.

Monet aiheet olivat tekijöille uusia. Erityisesti moninpeli tuotiin täysin uutena asiana. Tavoitteet etenivät projektissa aikataulun mukaisesti, vaikka alkuperäisiin lisättiin uusia tarpeita. Kehityksen edetessä simulaatiosovelluksen valmius pystyttiin visuaalisesti ja rakenteellisesti näkemään, mikä mahdollisti tavoitteiden kartoittamisen samanaikaisesti.

Projektissa opitut aiheet todettiin valistavaksi ja tulevaisuuteen valmistaviksi. Kaikki opittu hyödynnettiin myös projektin aikana, minkä kautta projektin rakennetta paranneltiin. Tiukan aikataulun takia tämä ei ollut aina mahdollista.

Projektin simulaatiosovellus otettiin kokonaisuudessa vastaan positiivisesti ja sen tyyppiselle kehitykselle nähtiin hyötyä pelastusalan toimintaa ja tulevaisuutta varten. Erityisesti mahdollisuus ympäristöjen näkemiseen ja tutustumiseen ennalta uskottiin auttavan operaatioiden ennakoinnissa. Lisäksi todettiin erilaisia uusia mahdollisuuksia ja ideoita simulaatioaiheeseen eri alojen kannalta.

Lähteet

Freepubg. 2021. Advantages and disadvantages of multiplayer online games. Viitattu 13.11.2022. Saatavissa https://www.easyfie.com/read-blog/16125_advantages-and-disadvantages-of-multiplayer-online-games.html

Mirror. 2021. Mirror Documentation. General. Viitattu 13.11.2022. Saatavissa https://mirror-networking.gitbook.io/docs/general

Mirror. 2022. Mirror Networking. Viitattu 13.11.2022. Saatavissa https://mirror-networking.gitbook.io/docs/

Photon Engine a. Photon Engine rakenne. Viitattu 13.11.2022. Saatavilla https://www.photonengine.com/en-US/Photon

Photon Engine b. Photon Realtime Intro. Viitattu 13.11.2022. Saatavissa https://doc.photonengine.com/en-us/realtime/current/getting-started/realtime-intro

Photon Engine c. Pun vs Bolt. Photon Engine. Viitattu 13.11.2022. Saatavissa https://doc.photonengine.com/en-us/bolt/current/reference/pun-vs-bolt

Photon Engine d. Photon Fusion Intro. Viitattu 13.11.2022. Saatavissa https://doc.photonengine.com/en-us/fusion/current/getting-started/fusion-intro

Photon Engine e. Photon Quantum Intro. Viitattu 13.11.2022. Saatavissa https://doc.photonengine.com/en-us/quantum/current/getting-started/quantum-intro

Photon Engine f. Photon Voice Intro. Viitattu 13.11.2022. Saatavissa https://doc.photonengine.com/en-us/voice/current/getting-started/voice-intro

Photon Engine g. Photon Chat. Viitattu 13.11.2022. Saatavissa https://www.photonengine.com/chat

Photon Engine h. PUN Introduction. Viitattu 20.11.2022. Saatavissa https://doc.photonengine.com/en-us/pun/current/getting-started/pun-intro

Photon Engine i. PUN Networked Objects. Viitattu 20.11.2022. Saatavissa https://doc.photonengine.com/en-us/pun/current/getting-started/feature-overview#networked_objects

Photon Engine j. PUN Player Networking. Viitattu 20.11.2022. Saatavissa https://doc.photonengine.com/en-us/pun/v1/demos-and-tutorials/pun-basics-tutorial/player-networking

Photon Engine k. PUN Instantiation. Viitattu 20.11.2022. Saatavissa https://doc.photonengine.com/en-us/pun/current/gameplay/instantiation

Photon Engine l. PUN MonoBehaviourPunCallbacks Class Reference. Viitattu 20.11.2022. Saatavissa https://doc-api.photonengine.com/en/pun/v2/class_photon_1_1_pun_1_1_mono_behaviour_pun_callbacks.html

Photon Engine m. PUN Callbacks. Viitattu 20.11.2022. Saatavissa https://doc.photonengine.com/en-us/pun/current/getting-started/feature-overview#callbacks

Photon Engine n. Synchronization and State. Viitattu 20.11.2022. Saatavissa https://doc.photonengine.com/en-us/pun/current/gameplay/synchronization-and-state

Photon Engine o. RPCs and Raise Event. Viitattu 20.11.2022. Saatavissa https://doc.photonengine.com/en-us/pun/current/gameplay/rpcsandraiseevent#shortcuts_for_rpc_names,%20Shortcuts%20For%20RPC%20Names

Salmi, T. 2021. Simulaation käsikirjoitus. Diasarja. Viitattu 30.11.2022. Ei saatavissa

SmartFoxServer a. Viitattu 13.11.2022. Saatavissa https://www.smartfoxserver.com

SmartFoxServer b. Overview. Viitattu 13.11.2022. Saatavissa https://www.smartfoxserver.com/overview

Unity3D. 2021. Testing multiplayer games locally. Viitattu 20.11.2022. Saatavissa https://docs-multiplayer.unity3d.com/netcode/current/tutorials/testing/testing_locally/index.html

Unity3D. 2022 a. Unet multiplayer. Viitattu 13.11.2022. Saatavissa https://docs.unity3d.com/Manual/UNet.html

Unity3D. 2022 b. About Netcode for GameObjects. Viitattu 13.11.2022. Saatavissa https://docs-multiplayer.unity3d.com/netcode/current/about

Wikipedia. Multiplayer video game. Viitattu 13.11.2022. Saatavissa https://en.wikipedia.org/wiki/Multiplayer_video_game