jamk

Unity Netcoden Toimivuus 3D-Moninpeleissä

Tero Björkman

Opinnäytetyö, AMK
Toukokuu 2022
Tietojenkäsittely ja tietoliikenne
Tieto- ja viestintätekniikan tutkinto-ohjelma (AMK)

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Björkman, Tero

Unity Netcoden Toimivuus 3D-Moninpeleissä

Jyväskylä: Jyväskylän ammattikorkeakoulu. **Toukokuu 2022**, 30 sivua

Tietojenkäsittely ja tietoliikenne. Tieto- ja viestintätekniikan tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Tehtävänä oli tutkia Unityn moninpelien kehitystyökalun, Netcoden, soveltuvuutta 3D-moninpelin tekemiseen pelinkehittäjän näkökulmasta. Tehtävä oli määrä ratkaista kehittämällä yksinkertainen kolmiulotteinen moninpeli, hyödyntäen mahdollisimman paljon internetistä löytyviä opetusohjelmia ja -videoita. Pelin tyyliksi valittiin ensimmäisen persoonan räiskintäpeli.

Aiempia tutkimustuloksia moninpelien suorituskyvystä löydettiin todella vähän. Siitä pääteltiin niiden olevan tärkeä lisäys pelinkehittäjän työkalupakkiin.

Moninpeli oli suhteellisen helppo kehittää Netcodea hyödyntäen. Joitakin ongelmia ilmeni peliä valmistaessa. Enimmät ongelmista johtuivat opetusohjelmien ja -videoiden vanhuudesta tai kokonaisuutena yhteensopimattomuudesta. Kaikki ongelmat olivat kuitenkin joko pääteltävissä aikaisemman Unity-kokemuksen pohjalta, tai ratkaistavissa Netcoden dokumentaatiota käyttäen. Muihin ratkaisemattomiin ongelmiin apua sai eri henkilöiltä internetin kautta.

Ensimmäinen tavoite oli selvittää, kuinka paljon yksinkertainen moninpeli vaatii resursseja palvelimeltaan. Toiseksi haluttiin selvittää, kuinka monta pelaajaa pystyvät yhtäaikaisesti pelaamaan palvelimella.

Tutkimusmenetelmäksi valittiin määrällinen tutkimus, sillä palvelimen kuormituksen mittaaminen tuottaa luonnollisesti numeroita, joilla tutkimuskohdetta voitiin helposti kuvailla. Mittaustulosten käsittely oli helppoa, sillä tulokset pystyttiin lataamaan sellaisenaan Microsoft Exceliin.

Tulokseksi saatiin tieto, että Netcode on oiva vaihtoehto moninpelin kehittämiseen. Yksinkertainen moninpeli ei kuormittanut palvelinta liiallisesti. Mittauksista kuitenkin huomattiin, että tietoliikenne palvelimella kasvoi enemmän, kuin palvelimen muu kuormitus pelaajamäärän kasvaessa.

Huomattiin myös, että todella suuren pelaajamäärän moninpeleissä tulee hyödyntää tietoliikenteen optimointia huomioimalla se pelin koodissa. Näin tietoliikenne ei kasva liialliseksi suurilla pelaajamäärillä.

Avainsanat (asiasanat)

Unity, Netcode, Moninpeli, Palvelin, Viive

Muut tiedot (salassa pidettävät liitteet)

jamk

Description

Björkman, Tero

Unity Netcode’s Functionality in 3D Multiplayer Games

Jyväskylä: JAMK University of Applied Sciences, May 2022, 30 pages

Information and communication technologies. Bachelor's Degree Programme in Information and Communications Technology. Bachelor’s thesis

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

The objectives were to research Unity’s multiplayer game development tool Netcode’s suitability for developing a 3D multiplayer game, from a game developer’s perspective. The objective was to be completed by developing a simple 3D multiplayer game utilizing as many online tutorials and tutorial videos as possible. The game was decided to be made in the style of a first-person shooter.

Very little prior research was found regarding multiplayer games’ performance. Due to this, it was decided that such research would be an important addition to a game developer’s toolbox.

A multiplayer game was relatively easy to develop using Netcode. Some problems arose while making the game. Most of the problems were caused by the old age, or incompatibility with one another, of the tutorials and tutorial videos. All the problems were solvable with some prior Unity experience, or by using Netcode’s documentation. Help for all other unsolved problems was available online from various persons.

The first goal was to find out what quantity of resources a simple multiplayer game demands from its server. The second goal was to find out how many players could play on the server simultaneously.

Quantitative research was chosen as the research method, since measuring the server naturally produced numbers with which to easily describe the researched issue. The measurements were also easy to process since they could be opened directly with Microsoft Excel.

As a result, it was discovered that Netcode is a good choice for developing a multiplayer game. A simple multiplayer game did not burden the server excessively. However, it was discovered from the measurements that data transfer on the server increased significantly as the number of players increased.

It was also discovered that multiplayer games with many players should utilize some form of code optimization for reducing data transfer. This way the data transfer will not grow to be too large.

Keywords/tags (subjects)

Unity, Netcode, Multiplayer game, Server, Latency

Miscellaneous (Confidential information)

Sisältö

Kuviot

Taulukot

Lyhenteet ja sanasto

2D	Kaksiulotteinen, esimerkiksi neliö
3D	Kolmiulotteinen, esimerkiksi kuutio
Asiakasohjelma	Tietokoneohjelma, joka tarvitsee toimiakseen palvelinta
Beetatestaus	Testi sovelluksen julkaisemattomasta versiosta
Funktio	Osa koodista, joka tekee halutun toiminnon
IP	Internet Protocol
Koodi	Kieli, jolla tietokonetta ohjeistetaan toimimaan
Moninpeli	Peli, jossa useampi kuin yksi pelaaja ovat toistensa kanssa vuorovaikutuksessa
Netcode	Unityn tarjoama työkalu moninpelien kehittämiselle
Palvelin	Tietokoneohjelma, joka palvelee toisia tietokoneohjelmia
Pilvi	Ulkoinen tietovarasto tai tietokone, johon yhdistetään internetin välityksellä
Prefab	Malli valmiista esineestä Unityssä
User interface	Käyttöliittymä, lyhennettynä UI
Viive	Aika, joka tiedolla kestää siirtyä paikasta toiseen

1 Johdanto

Videopelejä kehitetään yhä enemmän menneisiin vuosiin nähden. Jo pelkästään videopelialusta ja -kauppa Steam näki vuonna 2021 huikeat 11,775 uutta videopelijulkaisua (Steam Game Release Summary 2022). Kiitettävä osa näistä peleistä sisälsivät jonkinasteisia moninpeliominaisuuksia. Tarve hyvin toimivalle ja kattavalle moninpelien kehittämisalustalle on suuri ja kasvaa vuosi vuodelta.

Yksi suosituimpien pelinkehitysalustojen joukossa on Unity Technologiesin tarjoama Unity. Toinen, ehkä suosituin, kehitysalusta on Epic Gamesin kehittämä Unreal Engine. Näistä molemmat tarjoavat pelinkehittäjälle mahdollisuuden luoda myös moninpelejä. Pelinkehittäjälle on kuitenkin todella tärkeää käyttää hänen projektiinsa sopivaa pelinkehitysalustaa.

Unityllä kehitettyjä moninpelejä on julkaistu muutamia vuosien varrella. Näiden pelien määrä on kuitenkin suppea verrattaessa Unreal Enginellä kehitettyjen moninpelien määrään. Eteenkin suuressa suosiossa olevia, sekä suuria pelaajamääriä sisältäviä, Unityllä kehitettyjä moninpelejä on vaikea löytää. Unity-pelinkehitysalustan soveltuvuus moninpelien kehittämiseen on siis tärkeä tutkia.

Tutkimuksia Unityn moninpelityökaluihin liittyen löytyy kuitenkin todella vähän. Tätä opinnäytetyötä suunnitellessa koitettiin etsiä toisia samankaltaisia opinnäytetöitä. Kuitenkaan löydetyissä opinnäytetöissä, joissa käytettiin Unityä, tutkimus ei liittynyt moninpelien toimivuuteen.

1.1 Tutkimuksen tavoite

Tämän tutkimuksen tavoite on kartoittaa Unityn soveltuvuutta moninpelien kehittämiseen pelinkehittäjän silmissä. Sillä pelinkehittäjä ansaitsee elantonsa kehittämällä pelejä, joista pelaajat ovat valmiita myös maksamaan, hänen täytyy valita sopivat työkalut projekteihinsa. Näiden työkalujen vaihtaminen, esimerkiksi Unitystä Unreal Engineen, on erittäin vaikeaa, ellei mahdotonta kesken projektin. Lisäksi peli, sekä sen myötä kehittäjän palkkapäivä, saattavat myöhästyä, jos kehittäjän täytyy opetella käyttämään hänelle uutta pelinkehitysalustaa sen suosion takia. Siksi Unityn toimivuutta moninpeliprojekteihin täytyy tutkia.

Unity-pelinkehitysalustan tarjoama työkalu moninpelien kehittämiseen on Netcode. Tämä soveltuu sekä kaksiulotteisten, että kolmiulotteisten pelien luomiseen. Sillä suuri osa eteenkin suosituista peleistä 2020-luvulla ovat kolmiulotteisia, tässä tutkimuksessa perehdytään Netcoden toimivuuteen kolmiulotteisissa moninpeleissä.

1.2 Tutkimuksen tehtävät

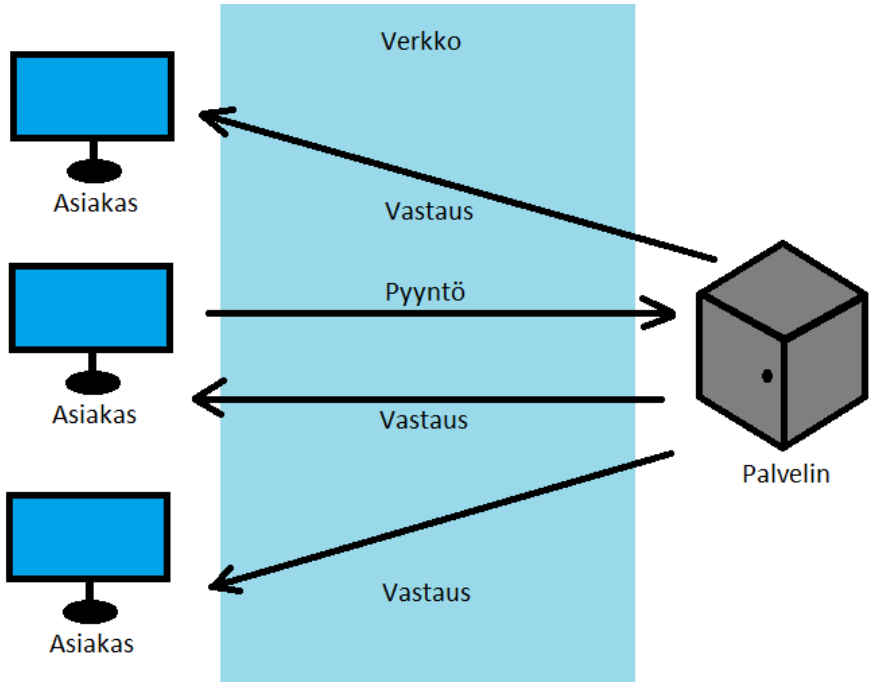
Tässä tutkimuksessa ensimmäinen tehtävä on kehittää uusi yksinkertainen kolmiulotteinen moninpeli Unityllä, käyttäen Netcodea. Pelin täytyy olla yksinkertainen, sillä kehittämiseen osallistuu vain yksi henkilö. Resurssit ovat siksi erittäin rajoitetut monimutkaisen tai visuaalisesti näyttävän moninpelin luomiseen. Lisäksi tutkimuksen on määrä perehtyä juuri Netcoden toimivuuteen, joka ulosrajaa näyttävyyden, sillä näyttävä peli vaatii resursseja enimmäkseen pelaajan käyttämältä tietokoneelta tai pelikonsolilta. Tämä ensimmäinen tehtävä pyritään toteuttamaan hyödyntämällä mahdollisimman paljon internetistä löytyviä opetusohjelmia ja -videoita Netcodeen liittyen.

Tutkimuksen toinen tehtävä on kuormittaa kehitettyä moninpeliä mahdollisimman suurella pelaajamäärällä. Usein suurin rajoittava tekijä moninpelin toimivuudessa on samanaikaisten pelaajien määrä. Pelaajamäärän kasvaessa yhteys pelissä olevien ohjelmien välillä saattaa tukkeutua liiallisesta tiedon liikkumisesta. Pelin toimivuus kärsii jyrkästi, elleivät peliin osallistuvat laitteet pysty pysymään yhteyksissä toisiensa kanssa.

2 Tietoperusta

2.1 Moninpeli

Moninpeli jakaantuu lukuisiin ohjelmiin ja niiden väliseen yhteydenpitoon. Näitä ohjelmia ovat vähintään palvelin, sekä yksi tai useampi asiakasohjelma (Ks. kuvio 1). Palvelin sijaitsee yleensä pelin ylläpitäjän tietokoneella tai pilvessä, mutta useissa peleissä yksi pelaajista voi toimia asiakasohjelman lisäksi myös itse palvelimena (Rintala 2019, 13). Asiakasohjelma sijaitsee yleensä pelaajan tietokoneella tai pelikonsolilla. Palvelin ja asiakasohjelmat pitävät yhteyttä toisiinsa tietoverkon yli. Tämä kanssakäyminen on paljolti rinnastettavissa esimerkiksi internetselaimen käyttöön.



Kuvio 1. Palvelimen ja asiakasohjelmien suhde toisiinsa

Asiakasohjelmat pyytävät palvelinta suorittamaan erilaisia käskyjä, esimerkiksi pelihahmon liikuttamista. Palvelin suorittaa pyydetyt käskyt ja palauttaa päivitetyn tilan pelistä takaisin asiakasohjelmille. Tämän jälkeen asiakasohjelmat päivittävät pelin tilanteen käyttäjille, jotka voivat syöttää uusia liikkumispyyntöjä.

Erillinen palvelin on sopiva valinta eteenkin suurien pelaajamäärien moninpeleille. Pelinkehittäjä voi siten suunnitella palvelimen suorituskyvyn vastaamaan pelille suunnitellun pelaajamäärän kuormitusta. Eteenkin tietokonepeleissä pelaajan tietokone voi olla mitä tahansa erittäin laadukkaan ja kaatopaikalta löydetyn väliltä. Konsolipeleissä tämä ongelma on paljon pienempi, sillä saman merkin pelikonsolit ovat tehty melkein samoista osista, yleensä vain konsolin sisältämän tallennustilan muuttuessa.

Kilpailuhenkisissä moninpeleissä käytetään lähes aina erillistä palvelinta. Täten on helpompi varmistaa, ettei kukaan pelaajista huijaa, sillä pelitilanne palvelimella on aina määräävä asiakasohjelmiin verrattuna. Kilpailuhengen säilyttämiseksi palvelimelle voi suhteellisen helposti ohjelmoida erilaisia tarkistusmenetelmiä asiakasohjelmilta saatujen käskyjen poikkeavuuksille.

2.2 Netcode

Netcode on Unity Technologiesin tarjoama työkalu moninpelien kehitykseen. Netcoden ensimmäinen julkaisu, versio 1.0.0-pre.2, julkaistiin alun perin 20.10.2021 (Netcode for GameObjects 2021). Netcodea edeltävä Unityn tarjoama vastaava työkalu oli nimeltään UNet, lyhennys nimestä Unity Networking (Anttonen 2019).

2.3 Palvelimen kuormitus

Pelaaja ei välttämättä itse huomaa moninpelin toiminnallisuudessa suurta eroa, jos pelaajamäärä kasvaa esimerkiksi kahdesta pelaajasta kolmeenkymmeneen. Hän saa yhä miltei saman määrän tietoa palvelimelta, lähettäen yhä saman määrän tietoa omalta asiakasohjelmaltaan. Tilanne on täysin toinen palvelimen puolella. Pelaajamäärän kasvaessa palvelin lähettää ja vastaanottaa moninkertaisen määrän tietoa, puhumattakaan enenevien käskyjen suorittamiseen vaaditusta kuormituksesta.

2.4 Viive

Suurin pelaajan kokema ero yksinpelin ja moninpelin välillä on viiveen suuruus. Viive esiintyy pelaajalle vasteajan eroina. Toisin sanoen, peli vaikuttaa vastaavan hitaammin pelaajan syöttämille toiminnoille viiveen kasvaessa (Bellomo 2021).

Viiveen suuruuteen vaikuttavat monet eri tekijät. Moninpeleissä näistä suurin on yleensä tietoverkon nopeus asiakasohjelman ja palvelimen välillä. Mikään tiedon liikkuminen ei kuitenkaan ole välitöntä. Viive siis kasvaa, aina hiiren ja tietokoneen välistä, palvelimen sisäisten johdinten yli, sekä verkkojohtojen sisällä. Tietoverkon sisäiseen viiveeseen vaikuttaa eteenkin asiakasohjelman ja palvelimen välillä olevien reitittimien määrä. Näitä reitittimiä löytyy sekä kaupunginosien, että maiden väliltä (Bellomo 2021).

2.5 Tutkimuksen rajaus

Tämä tutkimus toteutetaan pelinkehittäjän näkökulmasta. Siksi tutkimus rajataan käsittämään vain niitä asioita, joihin pelinkehittäjä voi eniten vaikuttaa pelinkehitysalustan valinnalla, sekä pelin sisältämällä koodilla. Tutkimusrajauksen ulkopuolella on lukematon määrä erilaisia muuttujia,

jotka vaikuttavat myös moninpelin toimivuuteen. Useimpiin näistä, kuten esimerkiksi asiakkaan käyttämän tietokoneen osiin, pelinkehittäjä ei pysty suoraan vaikuttamaan.

Ensimmäinen tutkittava asia on palvelimen kuormituksen määrä pelaajamäärän kasvaessa. Suuremmat pelaajamäärät vaativat luonnollisesti suorituskykyisempiä palvelimia. Pelinkehittäjä voi kasvattaa suurinta mahdollista pelaajamäärää sijoittamalla tehokkaampaan palvelimeen. On kuitenkin tärkeää, ettei pelinkehittäjä hanki 100 pelaajalle soveltuvaa palvelinta, jos hänen luoma moninpeli on suunniteltu vain 4 pelaajalle.

Toinen tutkittava asia on kaistanleveyden käyttö palvelimella pelaajamäärän kasvaessa. Kaistanleveyden käytön tarkastelun tarkoitus on laajalti sama, kuin palvelimen kuormituksen. Pelinkehittäjä voi sijoittaa tarvittaessa nopeampaan internet-yhteyteen palvelimelle, jos kaistanleveys ei riitä suunnitellulle pelaajamäärälle.

Kolmas tutkittava asia on palvelimen ja asiakasohjelman välillä oleva viive. Viiveen suuruuden vaihtelu pelaajamäärän kasvaessa antaa pelinkehittäjälle palautetta siitä, tarvitseeko pelin lähettämää ja vastaanottamaa tietoa vähentää. Esimerkkinä tästä voidaan käyttää hahmon liikkumista pelimaailmassa. Hahmo asiakasohjelmassa voi pyytää liikettä palvelimelta lähettämällä liikevektorin, eli pelaajan liikkeiden summan, jolloin palvelin laskee hahmon uuden sijainnin entisen sijainnin ja liikevektorin avulla. Toinen tapa liikkua voisi olla asiakasohjelman itsensä laskema uusi sijainti, jolloin se lähettää sijaintipyynnön palvelimelle hyväksyttäväksi. Tässä tapauksessa palvelin asettaa pelaajan uuden sijainnin palvelimella ja lähettää sijainnin asiakasohjelmille. Täten tietoa palvelimen ja asiakasohjelman välillä liikkuu vähemmän.

2.6 Aiempia tuloksia

Tutkimuksia pelaajamäärien suuruuden vaikutuksista palvelimella löytyy erittäin vähän varsinkin viime vuosilta. Ainoat tulokset liittyen pelaajamäärien vaikutukseen moninpelin toiminnallisuudessa ovat muutamat keskustelut Reddit-keskustelupalstoilla, eikä niitä tuloksia voida pitää luotettavana lähteenä tieteellisessä tutkimuksessa.

Tutkimusten vähäisyys johtuu todennäköisesti pelialan yrityksien salassapitosopimuksista. Moninpelit ovat varsin tuottava pelinkehityksen ala 2020-luvulla. Statista kertoo maailmanlaajuisen verkkopelaamisen ylittäneen 18 miljardin dollarin tuotot vuonna 2020 (Clement 2022). Tämän takia yritykset haluavat salata omaa teknologiaa, sekä tietotaitoa, kilpailijoiltaan. Samalla suuret pelialan yritykset ovat ainoita halukkaita sijoittajia tämänkaltaiseen tutkimukseen. Tutkimuksia kyllä suoritetaan varsinkin moninpelien Beetatestausvaiheessa, mutta nämä tulokset jäävät yrityksien sisäiseksi tiedoksi pelin viimeistelyä ja jatkokehitystä varten.

3 Tutkimusasetelma

Tämän tutkimuksen on tarkoitus selvittää pelinkehittäjille, onko Unityn tarjoama Netcode-työkalu hyvin soveltuva pohja moninpelin kehittämiselle. Määrällinen tutkimus soveltuu oivasti tämän kysymyksen vastaamiseen, sillä määrällisellä tutkimuksella kuvataan tutkittava kohde tilastojen ja numeroiden avulla (Määrällinen tutkimus 2015). Tutkimuksessa pyritään noudattamaan Euroopan unionin yleistä tietosuoja-asetusta, GDPR:ää.

Tutkimuksessa kehitetään uusi 3D-moninpeli internetistä löytyviä opetusohjelmia ja -videoita hyödyntäen. Tätä peliä testataan suurenevalla pelaajamäärällä järjestämällä testaustapahtuma. Testaustapahtumaan pyritään värväämään mahdollisimman paljon pelaajia sekä ystäväpiiristä, että videopelien kehittämiseen suuntautuneista sisällöntuottajista esimerkiksi YouTubesta. Tämä testaustapahtuma videoidaan siten, että palvelimen tietoja saadaan myös nauhalle pelin rinnalle. Peli rakennetaan siten, että palvelimella käynnistetyn pelin näkymän yläkulmassa näkyy luku palvelimelle liittyneistä pelaajista.

Ensimmäistä tutkimuksen kohdetta, palvelimen kuormituksen määrää kasvavalla pelaajamäärällä, tutkitaan lisäämällä testaustapahtuman videointiin näkymä palvelimen Windows Tehtävienhallintaikkunan Suorituskykyvälilehdestä. Täten videolta näkyy selkeästi pelaajamäärän vaikutus palvelimen suorittimen ja muistin kuormituksesta.

Toista tutkimuksen kohdetta, palvelimen kaistanleveyden käyttöä kasvavalla pelaajamäärällä, tarkkaillaan Windows Resurssienvalvontaikkunasta. Tämä ikkuna lisätään myös testaustapahtuman tallenteeseen. Resurssienvalvontaikkunasta näkyy selkeästi sovelluksen käyttämä lähetetyn ja vastaanotetun tietoliikenteen määrä.

Kolmatta tutkimuksen kohdetta, pelaajien kokemaa viivettä, tarkkaillaan kehittämällä projektin moninpeli siten, että peli tallentaa pilkulla erotettuun listatiedostoon pelaajien määrän ja viiveen sopivin väliajoin. Näin testaustapahtumassa saadut tiedot voidaan helposti siirtää Microsoft Exceliin kuvaajien luomista varten. Luoduista kuvaajista voidaan helposti nähdä pelaajamäärän kasvamisen vaikutus heidän kokemaan viiveeseen.

4 Moninpelin toteutus

Tässä tutkimuksessa kehitettiin yksinkertainen ensimmäisen persoonan 3D-räiskintäpeli käyttäen Unity-pelinkehitysalustaa. Tämän tyylin pelit tunnetaan yleisemmin first person shooter -genren peleinä. Peliin rakennettiin moninpeliominaisuudet hyödyntäen Unity Technologiesin tarjoamaa Netcode-työkalua. Tässä projektissa käytettiin Unityn versiota 2021.3.0f1.

Moninpelin kehitysosuus tässä tutkimuksessa toteutettiin seuraamalla internetistä löytyviä opetusohjelmia ja -videoita. Mitään yhtä opetusohjelmaa tämän projektin loppuun kehittämiseksi ei löytynyt. YouTubessa on monia 3D-räiskintäpelin kehittämisen opetusvideoita, mutta niistä harvat ovat räiskintämoninpelin opetusvideoita. Osa löytyneistä opetusohjelmista olivat myös vanhoja UNet-ohjelmia. Muutamat taas hyödynsivät Netcoden lisäksi muita moninpelinkehitystä helpottavia lisäosia. Sillä tässä tutkimuksessa oli määrä keskittyä Netcodeen, nämä opetusohjelmat eivät olleet sopivia vaihtoehtoja.

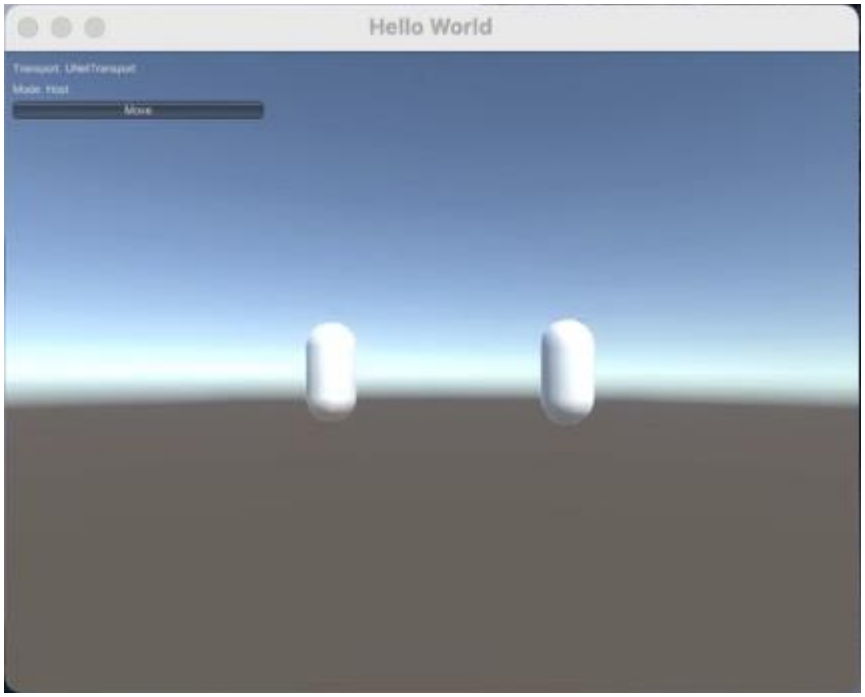
Peliä ei kuitenkaan ollut mahdollista rakentaa alusta loppuun pelkästään seuraten valmiita ohjelmia. Useissa vaiheissa kehitysprosessia opetusohjelmien lopputuloksia jouduttiin soveltamaan ja muokkaamaan yhteensopiviksi toistensa kanssa.

Valmis moninpeliprojekti kasvoi koodin luettavuuden kannalta suhteellisen sottaiseksi. Useassa paikoissa koodia sama funktio tekee monia toisiinsa suoraan liittymättömiä toimintoja. Sillä tämä projekti aloitettiin myöhään keväällä, aikaa ei kuitenkaan jäänyt toimivan koodin putsaamiseen.

4.1 Moninpelin perustukset

Pohjustus koko projektille luotiin seuraamalla Unityn tarjoamaa Netcode-opetusohjelmaa (Coughlin 2021). Tässä ohjelmassa luodaan toiminnallisuus palvelimen aloittamiselle, sekä palvelimelle

liittymiselle. Lisäksi luodaan pelaajalle prefab-malli, joka ilmestyy pelimaailmaan, kun pelaaja liittyy peliin (Ks. kuvio 2).



Kuvio 2. Kaksi pelaajaa liittyneenä pelimaailmaan (Testing the Movement on Hello World 2022, muokattu)

Pelaaja voi pyytää palvelinta liikuttamaan hänen hahmoaan Move-napin painalluksella, jolloin pelaajan hahmo siirtyy välittömästi satunnaiseen paikkaan pelimaailmassa. Move-napin toiminnallisuus koodissa on palvelimelle ja asiakasohjelmalle erilainen (Ks. Kuvio 3). Mikäli Move-nappia painetaan palvelimella, pelaajan paikaksi asetetaan suoraan uusi arvo. Muutoin sovellus lähettää palvelimelle kutsun siirtää pelaajaa, jolloin palvelin asettaa ja palauttaa uuden pelaajan paikan asiakassovelluksille.

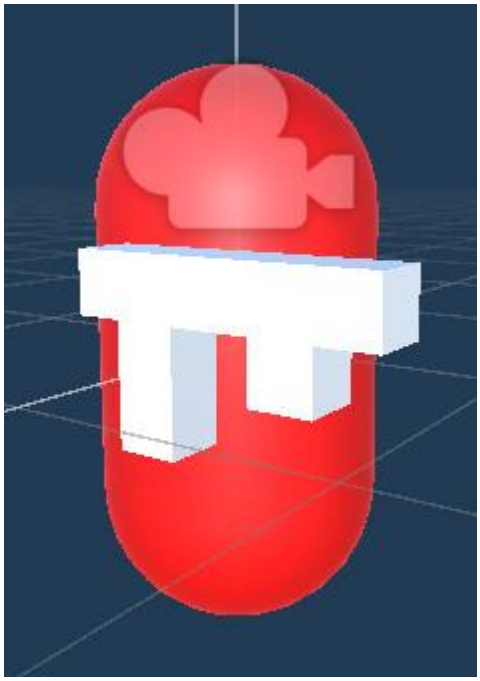
```
public void Move()
{
    if (NetworkManager.Singleton.IsServer)
    {
        var randomPosition = GetRandomPositionOnPlane();
        transform.position = randomPosition;
        Position.Value = randomPosition;
    }
    else
    {
        SubmitPositionRequestServerRpc();
    }
}
```

Kuvio 3. Move-funktion toiminnallisuus (Zurian 2022, muokattu)

Tämän opetusohjelman seuraamisessa ei esiintynyt suurempia ongelmia, sillä koko toimiva koodi kopioidaan suoraan projektiin. Lisäksi ohjelmassa on lisättynä videomateriaaleja, jotka ohjeistavat noudatettavat askeleet koodin ulkopuolella.

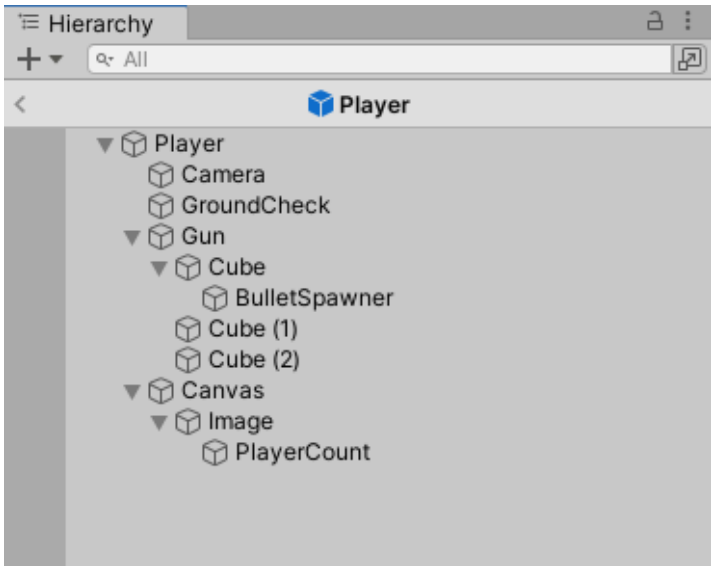
4.2 Pelaaja

Seuraavaksi muokattiin pelaajan prefab-mallia (Ks. kuvio 3). Pelaajalle lisättiin kamera, jonka kautta pelaaja näkee pelimaailman. Lisäksi pelaajalle luotiin asetta muistuttava muoto kuutioista, jotta pelaajan katsomissuunta on pääteltävissä toisille pelaajille. Pelkästä sylinteristä, josta pelaajan prefab koostui, ei katsomissuuntaa voi päätellä. Pelaajalle asetettiin kirkkaanpunainen väri, sillä pelaaja ei ensimmäisestä persoonasta näe oman hahmonsa ruumista, mutta väri auttaa pelaajia erottumaan pelimaailmassa.



Kuvio 4. Pelaajan päivitetty ulkomuoto

Pelaajalle lisättiin myös muita näkymättömiä osia. Näistä tärkeimmät ovat ”GroundCheck” ja ”BulletSpawner”, jotka esitellään osioissa 4.4 Liikkuminen, sekä 4.5 Räiskintä. Pelaajahahmon osaluettelo järjesteltiin loogiseksi (Ks. kuvio 5). Osahierarkia on tärkeä asettaa oikein, sillä Unityssä pelimaailman osat perivät sijaintinsa hierarkiassa ylempänä olevilta osilta. Lisäksi pelaajalle luotiin user interface -näkymä ”PlayerCount”, josta näkyy palvelimelle yhdistettyjen pelaajien lukumäärä.



Kuvio 5. Pelaajan osaluettelon hierarkia

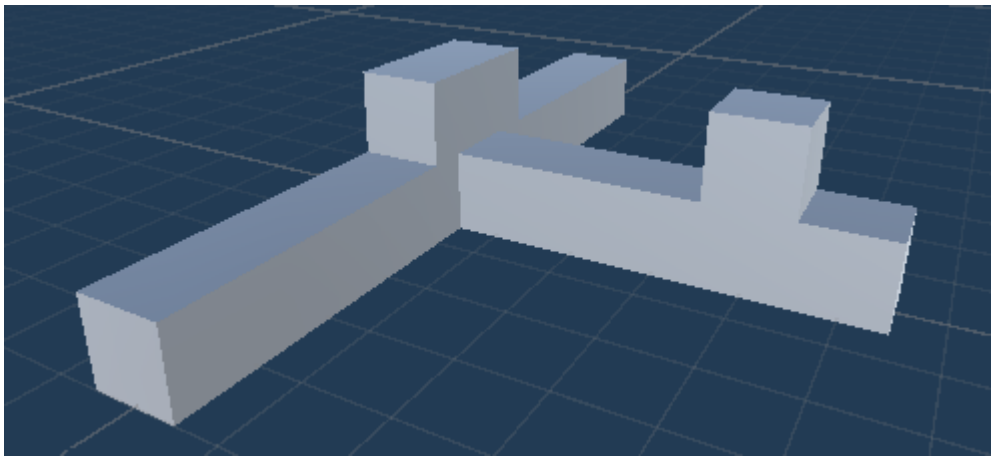
Pelaajalle asetettiin toiminnallisuus esineiden käytöstä poistamiseen uutta pelaajahahmoa luodessa maailmaan (Ks. kuvio 6). Esineiden käytöstä poistaminen täytyy tehdä, sillä muutoin jokaisen pelaajan kaikki osat ovat käytössä jokaisella asiakassovelluksella. Tämä loisi ongelmia eteenkin pelaajan kameran kanssa. Uuden pelaajan liittyessä peliin jokaisen pelaajan kameraksi päivittyisi uuden pelaajahahmon kamera, eivätkä he pystyisi näkemään peliä omasta kuvakulmastaan. Pelaajan liittyessä maailmaan, myös pelin aloitusnäytön kamera tuhottiin asiakassovelluksessa.

```
if (!IsLocalPlayer)
{
    camera.GetComponentInChildren<AudioListener>().enabled = false;
    camera.enabled = false;
    gameObject.layer = LayerMask.NameToLayer(remoteLayerName);
    gameObject.GetComponent<PlayerShoot>().enabled = false;
    //playerCount.enabled = false;
    Destroy(playerCount.GetComponentInParent<Image>());
    Destroy(playerCount);
}
```

Kuvio 6. Toiminnallisuus muiden pelaajien esineiden käytöstä poistamiseen

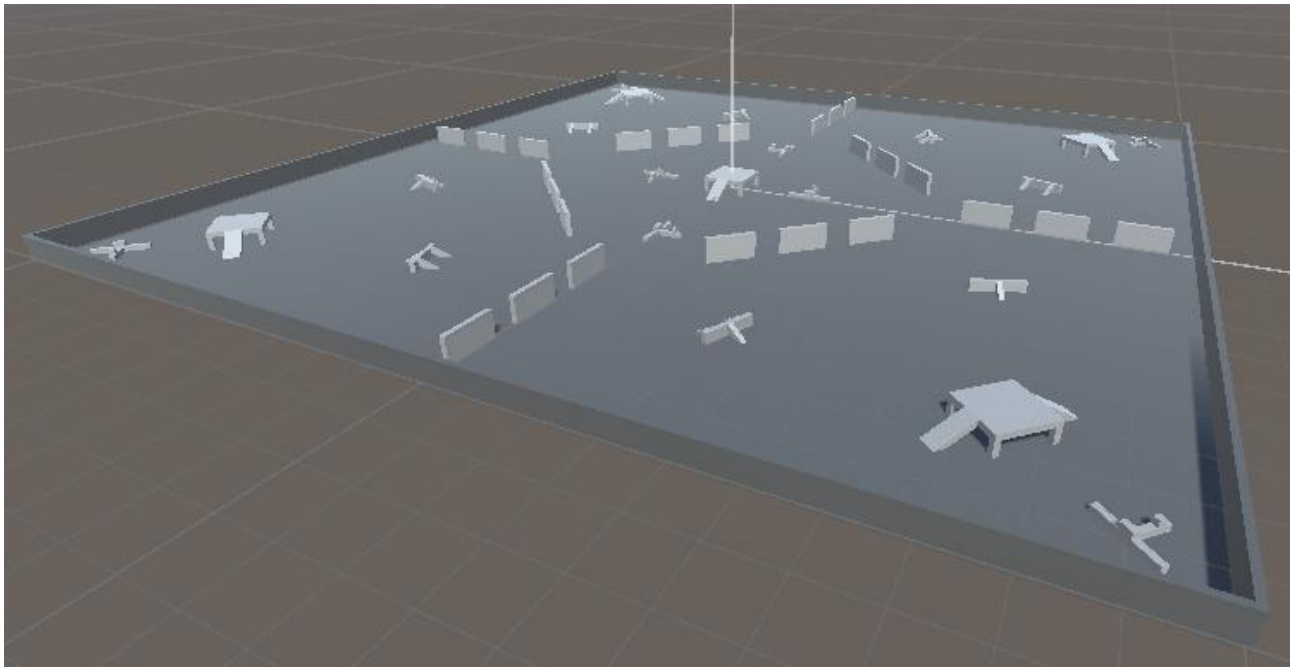
4.3 Pelimaailma

Projektille luotiin kohtuullisen suuri pelimaailma vastaamaan tavoitetta suuresta pelaajamäärästä. Tämä koostuu yhdestä suuresta tasosta, sekä erilaisista seinistä, esteistä ja rakennuksista. Kaikista esineistä, paitsi tasosta, luotiin prefab-mallit pelimaailman luomisen helpottamiseksi (Ks. kuvio 7).



Kuvio 7. Yksi pelimaailman esteiden prefab-malleista

Pelimaailman väreiksi valittiin harmaa uloimmille seinille ja lattiatasolle, sekä hieman vaaleampi muille esineille niiden erottuvuuden varmistamiseksi (Ks. kuvio 8). Uloimmat seinät asetettiin tarpeeksi korkeiksi, ettei pelaaja pystyisi ylittämään niitä hyppäämällä. Pelaajien näkyvyys toisilleen katsottiin tärkeäksi, joten pelimaailmasta tehtiin suhteellisen avoin.



Kuvio 8. Valmis pelimaailma

4.4 Liikkuminen

Pelaajan liikkumisen kehittäminen aloitettiin seuraamalla Brackeys-YouTubekanavan opetusvideota (FIRST PERSON MOVEMENT in Unity – FPS Controller 2019). Tässä videossa pelaajahahmolle luodaan koodit, joilla pelaajan liikettä ohjataan näppäimistön ja hiiren avulla. Opetusvideon lopussa pelaaja pystyy katsomaan ympärilleen, sekä liikkumaan ja hyppäämään.

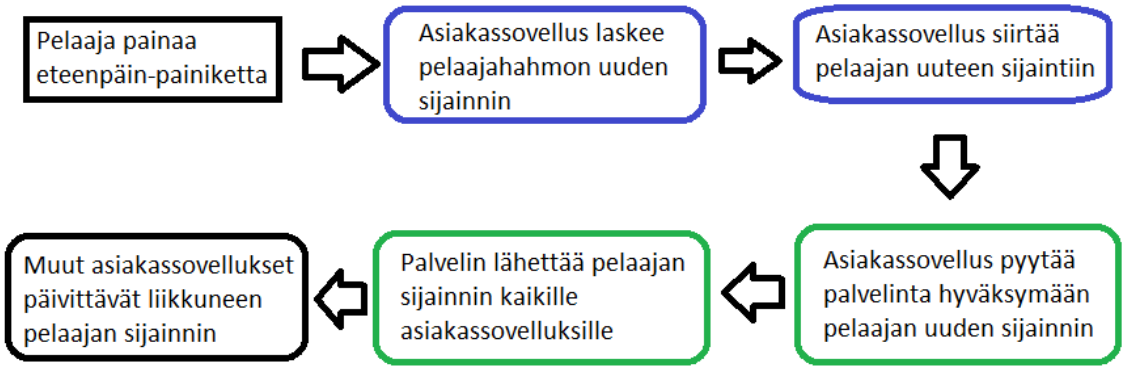
Alkuperäisessä opetusvideossa pelaajan kameraa ohjataan liikkumisen koodista erillisestä kooditiedostosta (Ks. kuvio 9). Kameran ohjaamisen koodi sisällytettiin kuitenkin liikkumisen kanssa samaan kooditiedostoon tässä projektissa. Tämä päätös helpotti ongelmien löytämisessä ja korjaamisessa, kun projektiin lisättiin moninpeliominaisuudet.

```
// Update is called once per frame
void Update()
{
    float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity * Time.deltaTime;
    float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity * Time.deltaTime;

    xRotation -= mouseY;
    xRotation = Mathf.Clamp(xRotation, -90f, 90f);

    transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);
    playerBody.Rotate(Vector3.up * mouseX);
}
```

Kuvio 9. Kameran liikuttaminen (FIRST PERSON MOVEMENT in Unity – FPS Controller 2019, muokattu)

Seuraavaksi kehitettiin liikkumisen toiminnallisuus moninpelissä. Tämä oli projektin vaikein vaihe, sillä aikaisempi opetusohjelma pelaajan liikkumiselle oli kehitetty yksinpelille. Se tarkoitti, että jokainen pelaaja pystyi liikkumaan moninpelissä oikein hyvin, mutta tämä liike ei välittynyt toisille pelaajille. Sen sijaan pelissä näkyvät muut pelaajat näyttivät pysyvän paikoillaan, vaikka he omassa asiakassovelluksessaan olivatkin liikkuneet.

Pelaajan liikkumisen moninpelitoiminnallisuus päätettiin tehdä minimoiden lähetettyä ja vastaanotettua tietoliikennettä palvelimen kanssa (Ks. kuvio 10). Asiakassovellukselle asetettiin tästä syystä myös toleranssi sille, kuinka paljon pelaajan sijainti sai poiketa palvelimella olevasta sijainnista. Tämä toleranssi pakotti pelaajan sijainniksi palvelimella olevan sijainnin, jos toleranssi ylitettiin kolmen sekunnin ajan.



Kuvio 10. Kaavio pelaajan liikkumisen toiminnallisuudesta moninpelissä

Pelaaja prefab-mallille luotiin myös toiminnallisuus liikkumisen poistamiselle käytöstä, ellei prefab-malli ollut luotu asiakassovelluksen omistamaksi. Jokainen asiakassovellus omistaa vain yhden kopion pelaajan prefab-mallista. Jollei toiminnallisuutta olisi rajattu, yhden pelaajan liikkuessa kaikki muutkin pelaajat noudattaisivat tämän pelaajan liikkeitä. Niille pelaajille, jotka eivät olleet asiakassovelluksen omistamia, kehitettiin sijainnin päivitysmetodi, joka siirsi kyseisen pelaajahahmon palvelimella olevaan sijaintiin.

4.5 Räiskintä

Räiskintätoiminnallisuuden luomisessa hyödynnettiin Brackeys-YouTubekanavan opetusvideota (Making a Multiplayer FPS in Unity (E06. Shooting) – uNet Tutorial 2015). Tämän videon lopputuloksena pelaaja saa mahdollisuuden ampua painamalla hiiren vasenta painiketta. Ampuminen on kuitenkin vielä videon lopussa näkymätöntä pelaajalle. Opetusvideo oli tehty vanhalle UNet-työkalulle, mutta sen kääntäminen toimimaan myös Netcodessa oli suhteellisen helppoa.

Opetusvideon lisäksi pelaajan ase asetettiin noudattamaan pelaajan kameran liikkeitä. Täten pelaajan ase ei pelkästään osoittaisi samaan suuntaan, johon pelaaja katsoo, vaan myös kääntyisi ylös ja alas pelaajan katseen mukana. Asiakassovelluksen omistaman pelaajan ase liitettiin suoraan kameran lapseksi pelaajan osaluettelon hierarkiassa, ottaen pois käytöstä aseen kääntyminen. Näin pelaaja itse näkee oman aseensa paikallaan sovelluksen oikeassa alakulmassa, kuten useissa räiskintäpeleissä on tapana.

Pelaajalle luotiin tähtäinristikko Brackeys-YouTubekanavan opetusvideon mukaan (Making a Multiplayer FPS in Unity (E10. Crosshair) – uNet Tutorial 2016). Videon lopputuloksena pelaaja saa kuvakulmansa keskelle user interface -kuvan, joka auttaa aseen tähtäämisessä. Tämä video kuuluu myös vanhaan UNet-opetusvideosarjaan. Tähtäimen luominen ei kuitenkaan käyttänyt UNetin toiminnallisuutta ja oli suoraan käytettävissä myös Netcodessa.

Pelaajan aseelle luotiin näkymätön osa ”BulletSpawner”. BulletSpawner sisältää toiminnallisuuden luoda uusi luoti maailmaan luodin prefab-mallin pohjalta (Ks. kuvio 11). Tämä osa on vastuussa luodin näkyvyydestä pelaajalle pelimaailmassa. BulletSpawner on pelaajan aseen lapsiesine, joka on kiinnitettynä aseen piipun päätyyn. Luodessa asiakassovelluksen omistamaa pelaajahahmoa, BulletSpawner siirretään hieman lähemmäksi kameraa tälle pelaajalle. Valinta siirrosta johtui luodin lähtösijainnista ampuvalle pelaajalle. Ilman siirtoa luoti näyttäisi lyhyen matkan räiskinnässä ilmestyvän liian alhaalta ja oikealta, eikä osuvan tähtäinristikon keskiosaan lentäessään eteenpäin.

```
public GameObject bulletPrefab;
public GameObject bulletSpawner;

3 references
public void SpawnBullet()
{
    //Debug.Log("Spawning bullet");
    //GameObject bullet = newbulletPrefab();
    Instantiate(bulletPrefab, bulletSpawner.transform.position, bulletSpawner.transform.rotation);
}
```

Kuvio 11. Luodin lisääminen pelimaailmaan

4.6 Yhdistäminen IP:n perusteella

Palvelimelle yhdistäminen IP:n perusteella saavutettiin hyödyntämällä SRCoder-YouTubekanavan opetusvideota (MLAPI Tutorial 21 Part 3 Making a Connect Address UI 2021). Videossa luodaan tekstikenttä, johon käyttäjä voi syöttää halutun palvelimen IP-osoitteen. Asiakassovellus koettaa yhdistää syötetyn IP-osoitteen omaavalle palvelimelle, kun pelaaja painaa ”Join” painiketta.

Tämä opetusvideo oli ainoa löytynyt ohje IP:n perusteella palvelimelle yhdistämisestä ja se oli luotu vanhalle UNet-työkalulle. Videon ohjeet olivat kuitenkin enimmäkseen käytettävissä myös Netcode-työkalulla. Ainoa ongelma oli IP-tekstikentän sisällön vaihtuessa IP:n asettaminen Unityn

uudelle Transport-osalle, joka vastaa muun muassa nettiyhteyden luomisen asetuksista (Ks. kuvio 12). Tähän ongelmaan löytyi kuitenkin apu Unity Multiplayer Networking Discord-keskustelupalstalta.

```
public void IPAddressChanged(string newAddress)
{
    ipAddress = newAddress;
    if (ipAddress == "")
    {
        ipAddress = "127.0.0.1";
    }
    //UnityEngine.Networking.NetworkTransport.
    NetworkManager.Singleton.GetComponent<UNetTransport>().ConnectAddress = ipAddress;
}
```

Kuvio 12. IP-osoitteen päivittäminen tekstikentästä

4.7 Pelaajamäärän ja viiveen tallennus

Pelaajamäärä ja viive tallennettiin pilkulla erotettu lista -muodossa tekstitiedostoon. Tämä helpotti kerätyn tiedon havainnollistamista Microsoft Excelissä, sillä tämän mallinen tiedosto on helppo kääntää ”.csv” tiedostoksi. Microsoft Excel osaa suoraan lukea csv-päätteisiä tiedostoja ja asetella ne oikein taulukkolaskentaohjelmaan.

Pelaajamäärä oli helppo saada tallennettua, sillä Netcodessa pelaajamäärän löytää suoraan Network Managerin tiedoista. Network Manager on se osa Netcoden koodista, joka ylläpitää verkkoyhteyttä ja tietoja pelaajista.

Viive oli hieman vaikeampi saada selville, sillä sen tarkasteluun ei suoraan löytynyt ohjetta internetistä. Ongelma oli kuitenkin ratkaistavissa lähettämällä muuttujassa palvelimen paikallinen aika asiakassovelluksille. Asiakassovellus palautti palvelimelta saadun ajan muuttujan takaisin palvelimelle, jolloin palvelin pystyi laskemaan viiveen vähentämällä asiakassovelluksen palauttama aika palvelimen ajantasaisesta ajasta (Ks. kuvio 13).

```
[ServerRpc]
1 reference
public void PingServerRpc(float serverPing)
{
    //Debug.Log("Two way ping: " + serverPing);
    if (!IsLocalPlayer)
    {
        Debug.Log("Current player count: " + NetworkManager.Singleton.ConnectedClientsIds.Count);
        serverPing = Time.time - serverPing;
        ThesisLogger.WriteString("Player count:," + NetworkManager.ConnectedClientsIds.Count + ",Ping," + serverPing.ToString());
    }
}
```

Kuvio 13. Viiveen laskeminen ja tiedon tallentamisen kutsu

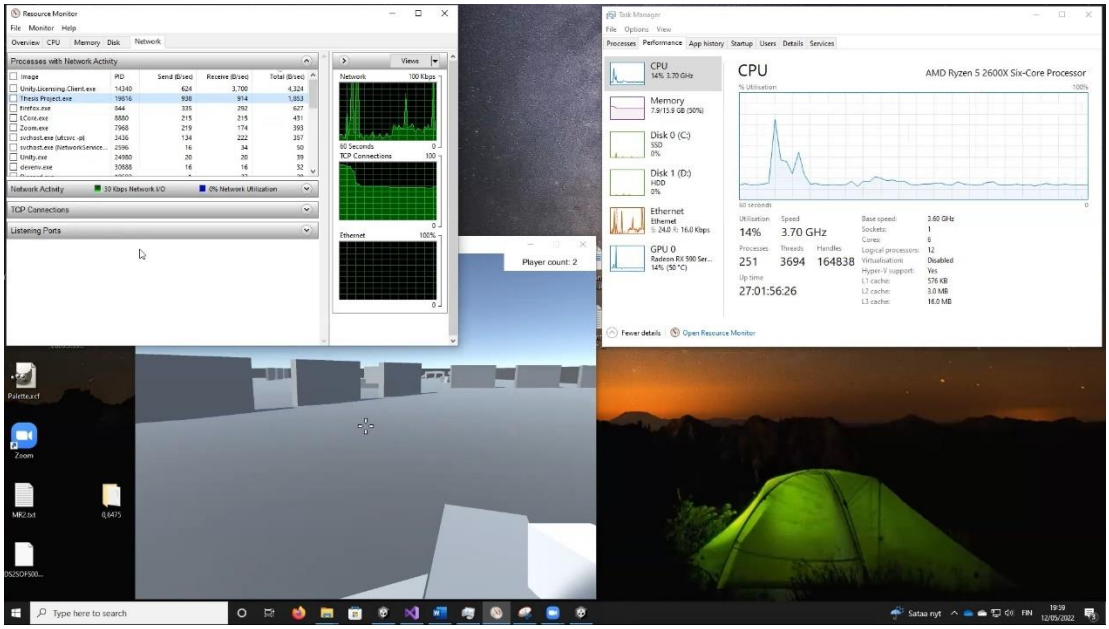
Itse tallennus suoritettiin Unityn StreamWriter-toiminnolla. Tämä toiminto vastaanottaa tekstijonon, sekä sille halutun tiedoston sijainnin tietokoneella. Sen jälkeen StreamWriter lisää syötetyn tekstin kyseisen tiedoston loppuun (Ks. kuvio 14).

```
public static void WriteString(string input)
{
    string path = "D:/ThesisPingLog/PingLog.txt";
    StreamWriter writer = new(path, true);
    writer.WriteLine(input);
    writer.Close();
}
```

Kuvio 14. StreamWriter-toiminnon käyttö

5 Tutkimuksen tulokset

Tutkimuksen testaustapahtumaan osallistui yhteensä 4 henkilöä, mukaan lukien palvelimen ylläpitäjä. Yhteensä moninpeliin yhdistyneitä asiakassovelluksia oli 13, sillä yksi tapahtuman henkilöistä yhdisti 10 asiakassovellusta palvelimelle hänen tietokoneeltaan. Tapahtuma videoitiin palvelimen näkymästä siten, että tallennuksessa näkyi myös palvelimen Resurssienvalvontaikkuna ja Tehtävienhallinnan Suorituskykyvälilehti (Ks. kuvio 15).



Kuvio 15. Testaustapahtuman näkymä palvelimella

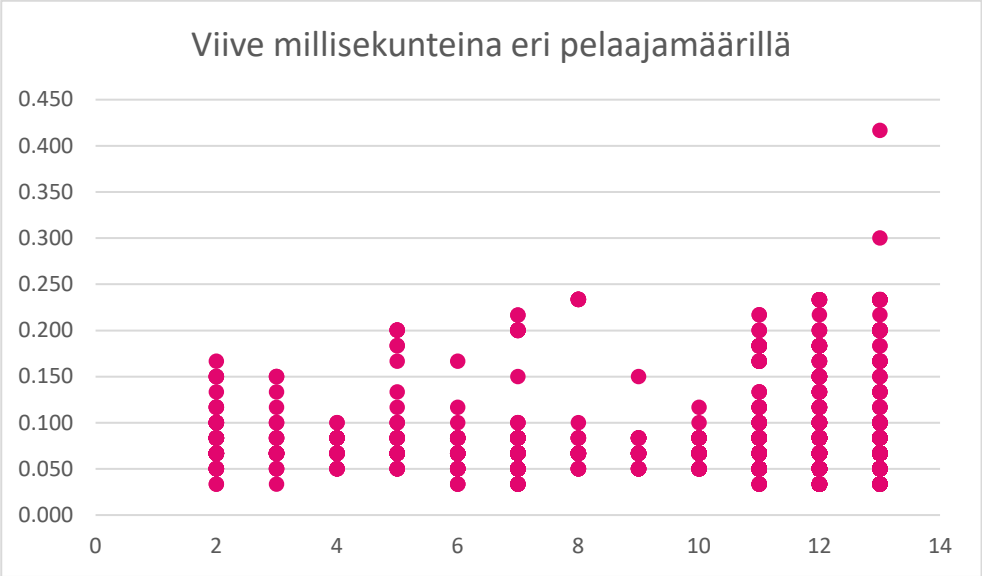
Palvelimena käytettiin yksityisessä käytössä olevaa Windows 10 -pöytätietokonetta. Testaustapahtuman ajaksi pöytäkoneelta suljettiin mahdollisimman paljon sovelluksia testin tulosten luotettavuuden varmistamiseksi. Palvelimella kuitenkin käytettiin samanaikaisesti Zoom-videokonferenssiohjelmaa, jolla testaustapahtuma videoitiin.

Taulukko 1. Testipalvelimen tiedot

Suoritin	AMD Ryzen 5 2600X Six-Core Processor 3.60 GHz
Muisti	16.0 GB 1067 MHz
Grafiikkasuoritin	Radeon RX 590 Series
Reititin	ASUS RT-N12E C1
Verkkoyhteys	89.3 Mbps lataus, 94.7 Mbps lähetys

5.1 Viiveen tulokset

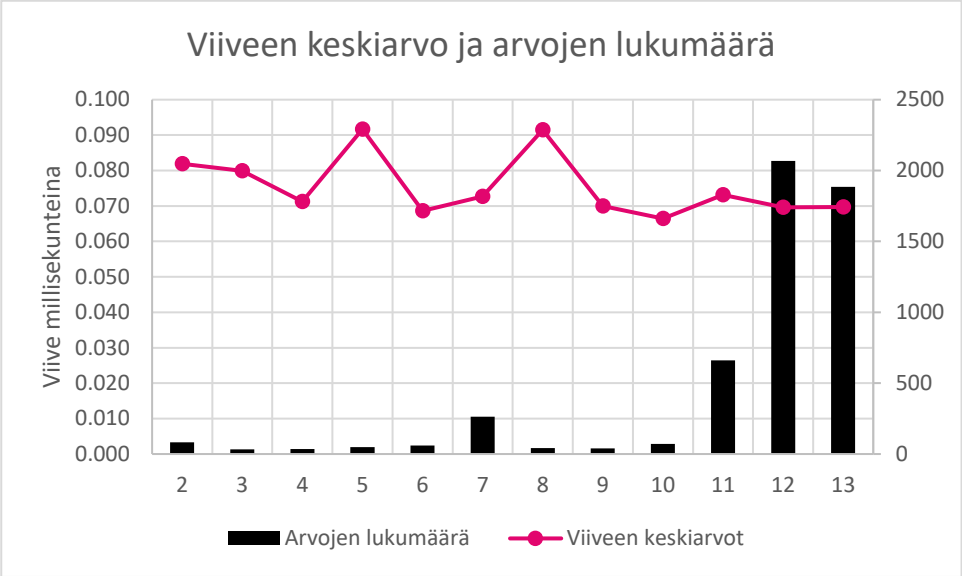
Testitapahtuman aikana tallennettiin paljon tietoa asiakassovellusten kokemasta viiveestä (Ks. kuvio 16). Viiveen tuloksia ei tallennettu vain yhden pelaajan ollessa liittynyt palvelimelle, sillä ensimmäinen pelaaja oli palvelimen ylläpitäjä, eikä hän olisi kokenut viivettä.



Kuvio 16. Testaustulokset viiveelle

Viiveen arvo pysyi koko testitapahtuman ajan melko luotettavasti alle 200 millisekunnin, muutamin poikkeuksin. Eriskummallista viiveen mittaustuloksissa on 2 ja 10 pelaajan kokemien viiveiden erot. 2 ja 10 pelaajalta saatiin miltei sama määrä testaustuloksia, mutta 10 pelaajan kokema viive vaikuttaa olevan paljon vakaampi ja lyhyempi. Tämä vakaus on kuitenkin selitettävissä sillä, että 10 pelaajalta viivettä kerättiin vähemmän aikaa. Saman tietomäärän kerääminen 10 pelaajalla tapahtuu 5 kertaa nopeammin, kuin 2 pelaajalla. Luonnollista heittoa viiveisiin ei siksi ehdi tapahtua yhtä paljoa. Viiveen maksimiarvot vaikuttavat kasvavan pelaajamäärän mukana.

Keskiarvo viiveille vaihteli kaikilla pelaajamäärillä 70 ja 90 millisekunnin välillä (Ks. kuvio 17). Viiveen keskiarvo vaikuttaa ensisilmäyksellä laskevan pelaajamäärän kasvaessa. Kuten viiveiden arvojen tuloksissa, keskiarvossa täytyy myös huomioida testaustulosten keräyksen aikaväli. Kaikista luotettavimmat keskiarvot viiveelle ovat 12 ja 13 pelaajan kanssa, sillä kyseiset pelaajamäärät olivat testitapahtumassa yhdistettyinä palvelimelle pisimmän aikaa.

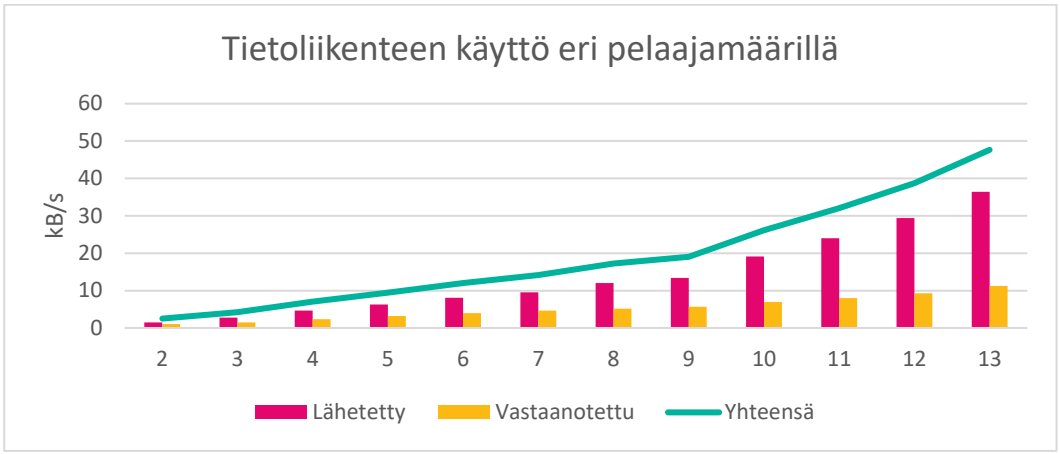


Kuvio 17. Viiveen analysoidut tulokset

5.2 Palvelimen kuormituksen tulokset

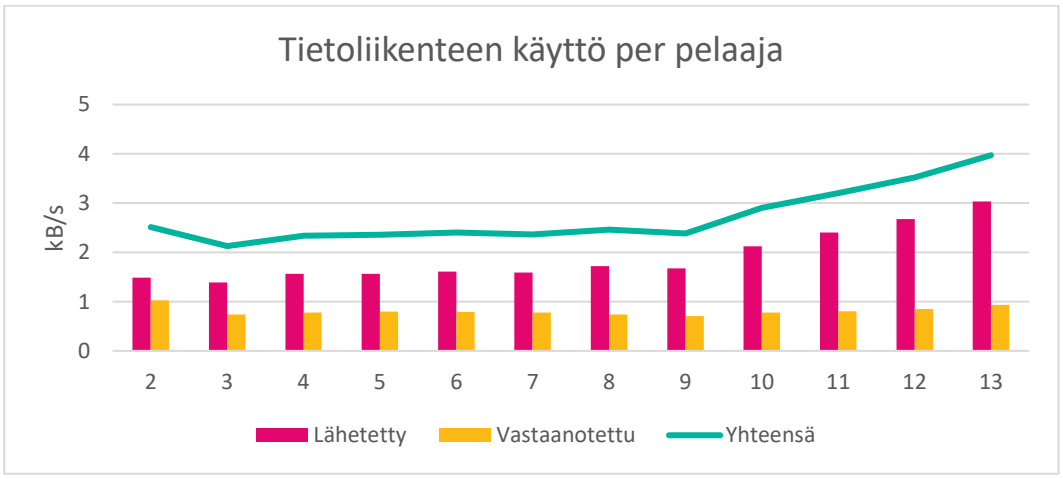
Testitapahtumasta tallennetusta videosta valittiin sattumanvarainen ajankohta jokaiselta pelaajamäärältä. Tästä ajankohdasta merkattiin ylös palvelimen tietoliikenteen, suorittimen, sekä muistin käyttö.

Tietoliikenteen käyttö kasvoi pelaajamäärän mukana (Ks. kuvio 18). Tietoliikenne vaikuttaa kasvavan lineaarisesti 2 ja 9 pelaajan välillä. 9 ja 13 pelaajan välillä kasvu nopeutuu, mutta pysyy lineaarisena. Tämä saattaa osittain johtua arvojen satunnaisesta valinnasta, kasvun ollessa todellisuudessa mahdollisesti eksponentiaalista. Erityistä huomiota täytyy kiinnittää lähetettyyn tietoliikenteeseen, sillä se kasvaa runsaasti nopeammin, kuin vastaanotettu tietoliikenne.



Kuvio 18. Tietoliikenteen käyttö

Microsoft Excelillä laskettiin tietoliikenteen käyttö per pelaaja (Ks. kuvio 19). Tulos saatiin jakamalla käytetty tietoliikenne pelaajien määrällä, josta on vähennetty palvelimen ylläpitäjä. Tietoliikenteen käyttö per pelaaja vaikuttaa pysyvän samana 9 pelaajaan asti, jolloin tapahtuu jonkinasteista kasvua lähetetyssä tietoliikenteessä aina uuden pelaajan liittyessä peliin.



Kuvio 19. Käsitellyt tulokset tietoliikenteestä

Pelaajamäärän kasvu muistin, sekä suorittimen käyttöön oli suhteellisen vähäistä. Muistin käyttö pysyi koko testitapahtuman aikana 7,9 gigatavun lukemassa. Muistin käyttö vaihteli vain hieman, noin 0,2 gigatavua molempiin suuntiin. Tätä vaihtelua kuitenkin esiintyi sekä 2:lla, että 13 pelaajalla. Suorittimen käyttö taas lisääntyi hieman 15 %:n kuormituksesta 20 %:n kuormitukseen pelaajamäärän kasvaessa.

6 Pohdinta

Unity Netcode vaikuttaa soveltuvan hyvin varsinkin alle 20 pelaajan 3D-moninpelien kehittämiseen. Netcoden käyttö ei ole liian vaikeaa, vaikka se olisikin pelinkehittäjän ensimmäinen moninpelien kehitystyökalu. Aikaisempi kokemus Unity-alustasta on kuitenkin suotavaa, sillä sopivia opetusohjelmia ja -videoita ei välttämättä löydy kaikentyyppisiin moninpeleihin. Useimmat ongelmien ratkaisut ovat kuitenkin pääteltävissä aiemmalla Unity-kokemuksella.

Pelinkehittäjän näkökulmasta, Netcode on mainio työkalu moninpelin kehittämiseen. Varsinkin palvelimen kannalta oleelliset, suorittimen ja muistin, kuormituksen mittaustulokset vaikuttavat pysyvän alhaisina pelaajamäärän kasvaessa. Kaistanleveyden käytön vähentämiseksi tulisi käyttää jonkinlaisia optimointimenetelmiä, jos pelinkehittäjä aikoo luoda vielä suuremman pelaajamäärän pelin. Näin palvelimiin sijoittamiseen vaadittava rahasumma pysyy vähäisenä.

Kerätyt tulokset Netcodella kehitetyn 3D-moninpelin toimivuudesta luultavimmin auttavat tulevaisuuden pelinkehittäjiä valitsemaan heille sopivan alustan. Oli tämä sitten Unity tai jokin muu. Vähintään tulokset antavat suuntaa tulevaisuuden 3D-moninpelien pelaajamäärille. Peli, jolla nämä tulokset kerättiin, on monella tapaa yksinkertaisin, mitä peliksi voi kutsua. Vaatimukset palvelimelle ovat todennäköisesti paljon suuremmat, kun peliin lisätään erilaisia toiminnallisuuksia.

Tutkimukset, jotka tulevaisuudessa tehdään palvelimen toimivuudesta moninpeleissä, tulisi suorittaa vieläkin suuremmalla pelaajamäärällä. Lisäksi tulisi hyödyntää työkaluja, jotka tallentaisivat myös tietoliikenteen, suorittimen, sekä muistin käytön tiedostoon sopivin väliajoin. Tällöin myös nämä tiedot olisivat helposti käsiteltävissä, sekä parantaisivat kerätyn tiedon luotettavuutta. Tulevaisuuden tutkimukset olisi myös suositeltavaa suorittaa täysin kyseiselle tutkimukselle tarkoitetulla laitteella, henkilökohtaisessa käytössä olevan tietokoneen sijaan. Tutkimukseen tulisi myös varata enemmän, kuin kaksi kuukautta aikaa.

Lähteet

Anttonen, M. 2019. Online multiplayer on mobile game. Opinnäytetyö, AMK. Turun ammattikorkeakoulu, Degree Programme in Information and Communications Technology. Viitattu 16.05.2022. https://urn.fi/URN:NBN:fi:amk-2019061216611

Bellomo, S. 2021. Latency and Packet Loss. Artikkeli Unity Technologiesin sivustolla. Viitattu 16.05.2022. https://docs-multiplayer.unity3d.com/netcode/current/learn/lagandpacketloss/index.html

Coughlin, B. 2021. Your First Networked Game "Hello World". Artikkeli Unity Technologiesin sivustolla. Viitattu 16.05.2022 https://docs-multiplayer.unity3d.com/netcode/current/tutorials/helloworld/helloworldintro

Clement, J. 2022. Online gaming – statistics & facts. Statista 09.05.2022. Viitattu 16.05.2022. https://www.statista.com/topics/1551/online-gaming/#dossierKeyfigures

FIRST PERSON MOVEMENT in Unity – FPS Controller. 2019. Brackeys. YouTube-videopalvelu. Viitattu 16.05.2022. https://www.youtube.com/watch?v=_QajrabyTJc

Making a Multiplayer FPS in Unity (E06. Shooting) – uNet Tutorial. 2015. Brackeys. YouTube-videopalvelu. Viitattu 16.05.2022. https://www.youtube.com/watch?v=QB2dt7kGlV8

Making a Multiplayer FPS in Unity (E10. Crosshair) – uNet Tutorial. 2016. Brackeys. YouTube-videopalvelu. Viitattu 16.05.2022. https://www.youtube.com/watch?v=_HeAaoLOvow

MLAPI Tutorial 21 Part 3 Making a Connect Address UI. 2021. SRCoder. YouTube-videopalvelu. Viitattu 16.05.2022. https://www.youtube.com/watch?v=HLpVVLj4uo0

Määrällinen tutkimus. 2015. Jyväskylän yliopisto. Viitattu 16.05.2022. https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/menetelmapolku/tutkimusstrategiat/maarallinen-tutkimus

Netcode for GameObjects. 2021. Päivitysmuistiinpanot GitHub-sivustolla. Unity Technologies. Viitattu 16.05.2021. https://github.com/Unity-Technologies/com.unity.netcode.gameobjects/releases

Rintala, T. 2019. Moninpelikehitys käyttäen Unity Unet -verkkotyökaluja. Opinnäytetyö, AMK. Tampereen ammattikorkeakoulu, Degree Programme in Business Information Systems. Viitattu 16.05.2022. https://urn.fi/URN:NBN:fi:amk-201905067783

Steam Game Release Summary. 2022. SteamDB. Viitattu 16.05.2022. https://steamdb.info/stats/releases/

Testing the Movement on Hello World. 2022. Unity Multiplayer Networking. YouTube-videopalvelu. Viitattu 16.05.2022. https://www.youtube.com/watch?v=2FTSAMbNXow

Zurian, V. 2022. Building on “Hello World”. Artikkeli Unity Technologiesin sivustolla. Viitattu 16.05.2022. https://docs-multiplayer.unity3d.com/netcode/current/tutorials/helloworld/hello-worldtwo