Moninpelin toteutus Unityllä

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus
kevät, 2023

Joona Heinonen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä
Tekijä Joona Heinonen Vuosi 2023
Työn nimi Moninpelin toteutus Unityllä
Ohjaaja Tommi Lahti

TIIVISTELMÄ

Tämä opinnäytetyö käsittelee Unity-pelimoottorin virallista Netcode for GameObjects -moninpelikirjastoa. Opinnäytetyön tarkoituksena oli saada selville, miten kirjasto mahdollistaa moninpelien kehittämisen Unityllä sekä miten sen käyttöönotto tapahtuu. Työn aikana luotiin demo, jonka avulla testattiin kirjaston sisältämiä toiminnallisuuksia kahden eri tietokoneen välillä.

Opinnäytetyö on tutkimuksellinen. Opinnäytetyön teoreettisessa osuudessa keskitytään pääsääntöisesti tutkimaan kirjaston sisältämiä toimintoja ja komponentteja moninpelien olennaisimpien toiminnallisuuksien mahdollistamiseen. Opinnäytetyön tietopohja koostuu enimmäkseen Unityn virallisen dokumentaation läpikäymisestä sekä kirjastosta luotujen ohjeiden tutkimisesta. Aineistoa tutkimalla kyetään havainnoimaan kirjaston tärkeimpiä osia, joita työssä tutkitaan. Tutkittuja osa-alueita hyödynnetään työn aikana rakennettavassa demossa.

Työn tuloksena huomattiin kirjaston käyttöönoton olevan yksinkertainen prosessi. Sen sisältämät komponentit ja toiminnot mahdollistavat moninpelikehitykselle tärkeät toiminnallisuudet ja niiden käyttö koettiin helpoksi. Kirjasto on vielä tuore, minkä takia vasta-alkajien voi olla vaikea hyödyntää kirjastoa. Aiemmin julkaistujen kirjastojen etuna on niiden käyttäjäkokemukset, jotka auttavat löytämään mahdollisesti esiintyviin ongelmiin ratkaisuja helpommin.

Avainsanat Unity, moninpelikirjasto, pelikehitys, demo
Sivut 24 sivua ja liitteitä 1 sivu

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Abstract

Author Joona Heinonen
Year 2023

Subject Unity Netcode for GameObjects

Supervisor Tommi Lahti

ABSTRACT

The purpose of the thesis was to explore Unity's new Netcode for GameObjects networking library and to find out how it enables multiplayer game development in the Unity game engine. The thesis sought to study different features of the library, which were then used in the demo that tested different components and functionalities included in the library.

The thesis is theoretical. The primary research method for the information included in the theoretical part of the thesis was to study Unity's official documentation of the library and tutorials made on the subject. The theoretical part focuses on the library's sections which are used for the main functionalities in multiplayer video games. The central subjects, which are explained in the theoretical part, are then tested in the demo created in the practical part of the thesis.

Based on the research, enabling the library is a simple process. The components and features included in the library are easy to access and they enable important functionalities used in multiplayer games. The library is still quite new, which might make it difficult to access for newcomers. Compared to other previously released libraries, there is not as much experience and solutions shared online by other users of the library.

Keywords Unity, multiplayer library, game development, demo
Pages 24 pages and appendices 1 page

Sanasto

Netcode	Verkkologiikkaa mahdollistavat kirjaston osat
Peliobjekti	Objekti pelissä, johon lisätään komponentti suorittamaan toiminnallisuuksia
Kirjasto	Ohjelmakokoelma
Komponentti	Toiminnallisuus, joka liitetään peliobjektiin
Skripti	Ohjelmoitu komponentti, joka liitetään objekteihin
Protokolla	Yhteyden ja tiedonsiirron mahdollistava käytäntö
Attribuutti	Olion muuttuja, jolle voidaan antaa arvo
Asiakasohjelma	Palvelimella oleva yksittäinen pelaaja
Prefab	Valmiiksi luotu peliobjekti, jota voidaan käyttää helposti uudestaan
Latenssi	Pelaajien ja verkon välillä oleva viive
Isännöity palvelin	Palvelin jota yksi pelaajista ylläpitää

Sisällys

Kuvat, ohjelmakoodit ja taulukot

Liitteet

1 Johdanto

Moninpelit mahdollistavat paljon. Rento hauskanpito ystävien kanssa, uusiin ihmisiin tutustuminen moninpelien välityksellä, sekä turnaukset, joissa pelataan jopa miljoonien voittosummista ovat vain pintaraapaisu moninpelien maailmaan.

Vaikkei minulla itselläni ollut opintoja edeltävää kokemusta videopelien kehittämisestä, olen aina ollut erittäin kiinnostunut asiasta videopeliharrastuneisuuden ansiosta. Opinnoissa suorittamani Unity-kurssin aikana tuli esille moninpelien toteuttaminen kyseisellä pelimoottorilla ja Unityn tuotantovalmis Netcode for GameObjects -kirjasto herätti mielenkiintoni.

Opinnäytetyössäni tulen kertomaan hieman yleistä tietoa Unitystä sekä mainitsemaan kolmansien osapuolien kehittämiä aikaisemmin julkaistuja moninpelikirjastoja. Pääosassa tulee olemaan virallisen kirjaston tarjoamat ratkaisut moninpelien erilaisten toimintojen mahdollistamiseen.

Työn käytännön osuudessa tullaan käymään vaihe vaiheelta läpi kirjaston sisältämiä toiminnallisuuksia hyödyntävä demo, joka toimii lähiverkkoyhteydellä kahden eri tietokoneen välillä.

Uuden kirjaston tutkiminen tulee auttamaan moninpelien luomisesta kiinnostuneita uusia kehittäjiä selvittämään, mitä kirjasto mahdollistaa sekä miten se toimii käytännössä.

Opinnäytetyön aikana pyritään vastaamaan kysymyksiin:

1. Miten Netcode for GameObjects mahdollistaa moninpelin tekemisen?

2. Kuinka Netcode for GameObjects otetaan käyttöön?

3. Miten kirjaston sisällön hyödyntäminen onnistuu vasta-alkajalta?

2 Unity

Unity on Unity Technologies -yrityksen vuonna 2005 julkaisema pelimoottori, jota käytetään 2D ja 3D pelien kehittämiseen. Pelimoottori on apuna niin kokeneille kehittäjille, kuin vasta-alkajille. Se tarjoaa laajat työkalut pelien kehittämiseen eri alustoille kuten tietokoneille, mobiililaitteille sekä eri pelikonsoleille. (Schardon, 2023)

2.1 Skriptaus

Unityllä kehittäessä voidaan skriptien avulla määrittää erilaisten objektien toiminnallisuuksia. Skripteissä voidaan määrittää luokkia, sekä asettaa erilaisia arvoja ja funktioita, joita objektien halutaan omaavan. Unityssä skriptit liitetään peliobjekteihin, jotta ne saisivat skripteissä asetetut ominaisuudet käyttöönsä. Skriptit ovat omia tiedostojaan, joissa hyödynnetään tiettyä ohjelmointikieltä ominaisuuksien luomiseen. (Unity, n.d.-a)

Unityssä käytetään ohjelmakoodissa 1 näkyvää C#-ohjelmointikieltä, jota Unity tukee oletuksena (Unity, 2023-a). C# on Microsoftin kehittämä olio-ohjelmointikieli, joka toimii .NET-arkkitehtuurin avulla. .NET sisältää C#-kehitykseen tarvittavat kirjastot, sekä ajoympäristön sillä luoduille ohjelmille. (Microsoft, 2022)

Ohjelmakoodi 1 Unityssä luotu C#-skripti ilman muokkauksia.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TestScript : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

2.2 Kolmansien osapuolien kirjastot

Kolmansien osapuolien luomat kirjasto ovat mahdollistaneet moninpelien luomisen Unityllä ennen Unityn virallista Netcode for GameObjects -kirjastoa. Kirjastojen välillä on eroja ja eri kirjastot sopivat tietynlaisiin peleihin paremmin kuin toiset. Kirjaston valitsemiseen vaikuttavia seikkoja ovat muun muassa pelaajien määrä, skaalautuvuus, sekä pelin tyyppi. (House, 2020)

Kolmansien osapuolien moninpelikirjastoja on monia erilaisia. Yksi näistä on Mirror, joka perustuu Unityn kehittämään jo vanhentuneeseen UNet:iin. Mirror kehitettiin korjaamalla UNet:n ohjelmointivirheitä, sekä parantamalla skaalautuvuutta. Tämä mahdollisti teknologian käytön laajemmissa moninpeleissä. (Mirror, n.d.)

Photon PUN (Photon Unity Networking) on ohjelmistopaketti, joka perustuu kolmeen ohjelmointirajapintaan. Ne sisältävät PUN:n koodin, jossa on Unityn puolella käsiteltävät ominaisuudet, logiikka, joka hoitaa verkon ja yhteyden asiat, sekä DLL-tiedostot. (Photon, n.d.-a)

Photon on myös julkaissut vuonna 2021 Fusion-nimisen ratkaisun moninpelikehitystä varten. Se kehitettiin korvaamaan yrityksen vanhempi teknologia ja se tarjoaakin huomattavasti enemmän toiminnallisuuksia kuin edeltäjänsä. (Photon, n.d.-b)

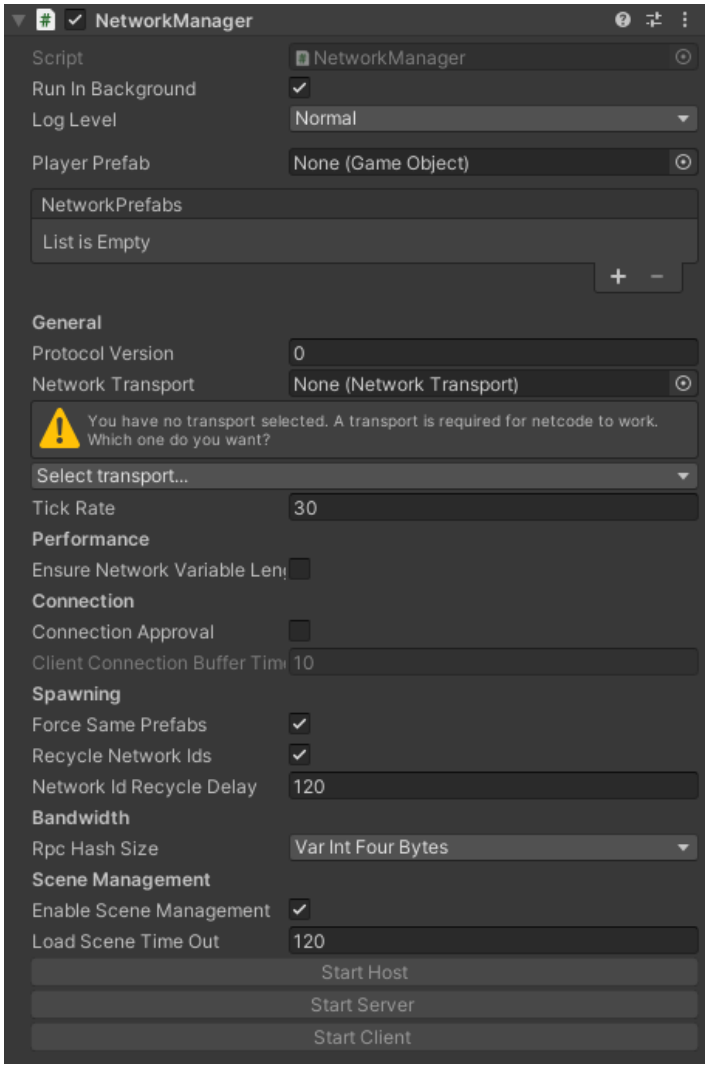
3 Netcode for GameObjects -kirjasto

Unityn virallisen Netcode for GameObjects -moninpelikirjaston tuotantovalmis versio julkaistiin syyskuussa 2022 (Unity, n.d.-b). Kirjasto korvasi Unityn aiemman MLAPI-kirjaston, jota ei enää Unityn dokumentaation mukaan tulla ylläpitämään tai päivittämään. Unity on jopa kehittänyt työkalun, jolla vanhat MLAPI:n avulla luodut projektit voidaan päivittää toimimaan Netcode for GameObjects -kirjaston kanssa. (Unity, 2023-b) Kirjasto mahdollistaa kehittämisen miltei kaikille alustoille, paitsi verkkoselaimilla toimivalle WebGL:lle (Unity, 2023-c). Unity itsessään ei sisällä kirjastoa valmiiksi, vaan se täytyy lisätä moninpeliprojekteihin erikseen.

3.1 NetworkManager

NetworkManager on Netcode for GameObjects-kirjaston toimivuuden kannalta pakollinen komponentti jokaiseen kirjastolla rakennettuun moninpeliprojektiin (Unity, 2023-d). Projektissa saa olla vain yksi NetworkManager, sillä se toimii pääkomponenttina, joka käsittelee miltei kaikki moninpeleihin liittyvät toiminnot. Komponentin kautta voidaan muuttaa pelin yhteyteen liittyviä asetuksia, kuten miten yhteys luodaan ja miten palvelin toimii. Kuvassa 1 esiintyvä NetworkManager sisältää myös NetworkPrefabs-listan, johon lisätään kaikki peliobjektit, joita halutaan käyttää verkossa. (Code Monkey, 2022)

Kuva 1 NetworkManager-komponentti vakioasetuksilla (Unity).

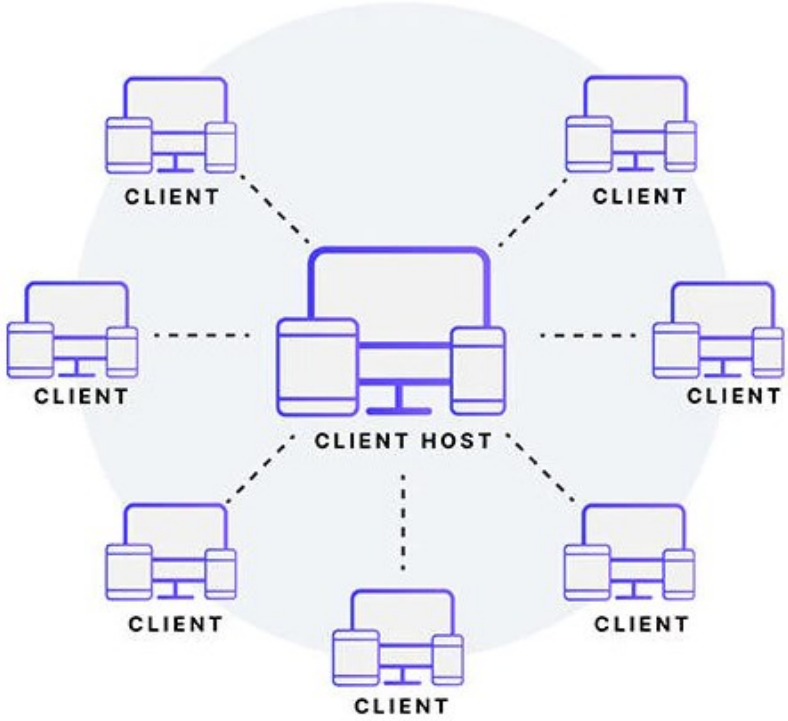


3.2 Palvelimen toiminta

Moninpeliä luodessa täytyy ottaa huomioon, miten pelin palvelinpuoli halutaan toteuttaa. NetworkManager-komponentin kautta voidaan joko käynnistää erillinen palvelin (server), isäntäpalvelin (host), tai pelkkä asiakasohjelma (client).

Kuten kuvassa 2 nähdään, isännöity (client-hosted) palvelin toimii sillä periaatteella, että yksi pelin pelaajista toimii palvelimen isäntänä, jolloin muut pelaajat liittyvät kyseiseen isäntäverkkoon, eikä erillistä palvelinta tarvita (Unity, n.d.-c). Tällöin NetworkManager käyttää isännän paikallista IP-osoitetta palvelimen ylläpitoon (Code Monkey, 2022). Erillistä palvelinta (dedicated server) käyttämällä pelin toimivuus ei ole sidottu yhteen pelaajista, jolloin pelaajat liittyvät suoraan asiakasohjelmillaan pelipalvelimelle.

Kuva 2 Isäntäpalvelin ja asiakasohjelmat (Unity, n.d.-c).



Palvelinmallin valintaan vaikuttaa käyttötarve ja molemmissa malleissa on omat etunsa. Taulukossa 1 nähdään osa palvelinmalliin vaikuttavista avaintekijöistä. (Unity, n.d.-c).

Taulukko 1 Vertailua palvelinmallien välillä (Unity, n.d.-c).

Avaintekijä	**Isännöity palvelin**	**Erillinen palvelin**
Latenssi	Riippuu isännän yhteyden laadusta	Tyypillisesti matala
Pelaajien määrä	n. 10 pelaajaa	Korkea
Kustannukset	Yleisesti ei ylimääräisiä kustannuksia	Vaihtelee palvelimien ylläpitopalveluiden mukaan
Huijaukseneston toteutus	Vaikea	Helppo

3.3 NetworkObject

NetworkObject on Network for GameObjects-kirjaston peliobjektien pakollinen komponentti, joka lisätään jokaiseen peliobjektiin, jotta ne saadaan toimimaan moninpeleissä. Komponentti yksilöi peliobjektit, jolloin jokainen peliobjekti saa oman tunnisteensa (NetworkObjectId) kun ne otetaan käyttöön palvelimella. Tunnisteen avulla muut komponentit osaavat kohdistaa vaadittavat toiminnallisuudet peliobjektiin, silloin kun niitä halutaan käyttää. (Unity, 2023-e)

3.4 NetworkBehaviour

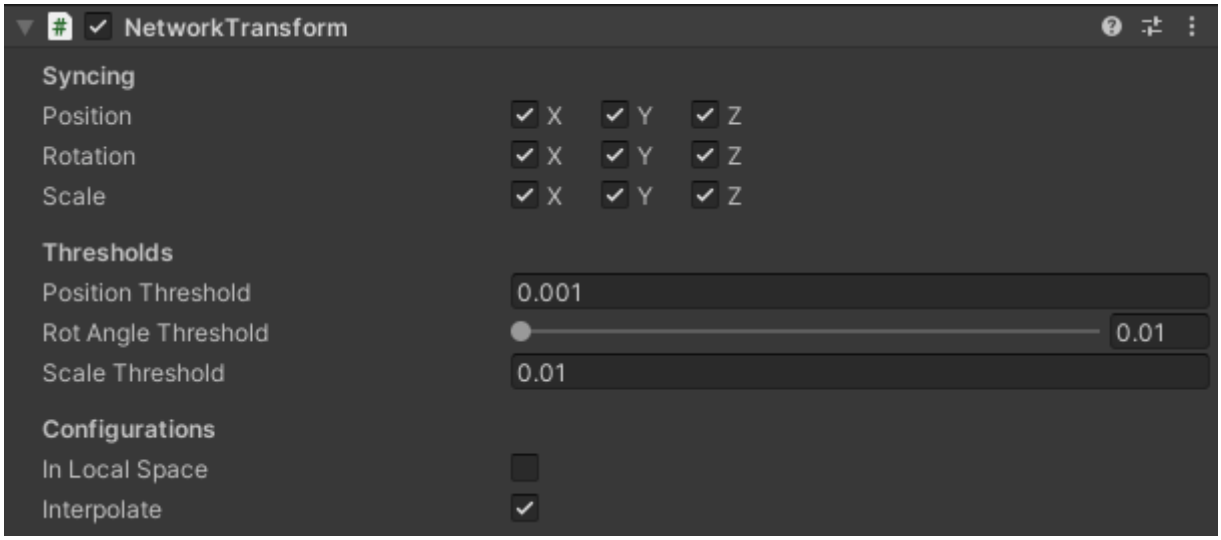
NetworkBehaviour-luokka on toinen pakollisista komponenteista moninpeliobjektien käyttöönotossa. Sitä käytetään skriptien puolella korvaamaan sen perimä MonoBehaviour-luokka, jolloin peliobjektit, joihin kyseinen skripti lisätään saavat NetworkBehaviour-luokan sisältämät ominaisuudet itselleen. Jos NetworkObject-komponenttia ei ole lisätty peliobjektiin ennen NetworkBehaviour-luokkaa perivää skriptiä, Unity lisää puuttuvan komponentin peliobjektiin automaattisesti. Yhdessä objektissa voi olla monta NetworkBehaviour-luokan skriptiä, jotka suorittavat eri toiminnallisuuksia. (Unity, 2023-f)

NetworkBehaviour-luokan perivään skriptiin voidaan eritellä peliobjektien toiminnallisuuksia niin, että vain halutut objektit tekevät niille osoitetut toiminnallisuudet. Esimerkiksi jos kyseessä on pelaajien hahmoja liikuttava skripti ja yksi pelaajista haluaa liikuttaa omaa pelihahmoaan, muut pelissä olevat hahmot eivät liiku. (Code Monkey, 2022)

3.5 NetworkTransform

NetworkTransform-komponenttia käytetään peliobjektien sijainnin, koon, sekä niiden tekemien käännösten synkronointiin palvelimella. Komponentin säädöt (Kuva 3) mahdollistavat tiettyjen akseleiden muutosten synkronoinnin, jolloin mahdollisesti tarpeettomien akseleiden muutosten synkronointi voidaan jättää pois. Pelin toiminnallisuuksille turhien arvojen synkronointi vaikuttaa pelin suorituskykyyn, koska synkronointi vaatii prosessointikykyä sekä hidastaa tiedonsiirtonopeutta. (Unity, 2023-g)

Kuva 3 NetworkTransform-komponentin asetukset (Unity).



NetworkTransform-komponenttia käytetään silloin, kun palvelimella tehdyt muutokset halutaan synkronoida näkymään asiakasohjelmilla. Jos halutaan, että peliobjektin muutos tehdään asiakasohjelmalla, käytetään ClientNetworkTransform-komponenttia, joka on täysin samanlainen kuin NetworkTransform. Komponentti ei kuulu oletuksena Netcode for GameObjects-kirjastoon. Se on osa Multiplayer Samples Utilities -pakkausta, joka lisätään tarvittaessa projektiin erikseen. (Unity, 2023-g)

3.6 Remote Procedure Call

Remote Procedure Call (RPC) on tietoliikenneprotokolla, joka synkronoi tietoa palvelimen ja asiakasohjelmien välillä. Sitä kutsumalla pystytään suorittamaan toimintoja objekteissa, jotka eivät kuulu samaan suoritettavaan tiedostoon. RPC:itä voi kutsua joko asiakasohjelmalta (ServerRpc), tai palvelimelta (ClientRpc). (Unity, 2022-a)

3.6.1 ServerRpc

ServerRpc lähettää tiedon asiakasohjelmalta palvelimelle, jolloin asiakasohjelman puolella tehty toiminto muuttuu verkossa tehdyksi toiminnoksi. ServerRpc muodostetaan NetworkBehaviour-luokan perivän skriptin puolella metodina, joka luodaan lisäämällä metodin loppuun ServerRpc-pääte. Metodi vaatii myös attribuutin [ServerRpc] metodin

yläpuolelle, jotta se tunnistettaisiin RPC-metodina. Ohjelmakoodissa 2 nähdään esimerkki ServerRpc:n pohjustuksesta. (Unity, 2022-b)

Ohjelmakoodi 2 ServerRpc luotuna skriptissä.

```
public class TestScript : NetworkBehaviour
{

    [ServerRpc]
    0 references
    public void TestServerRpc (int test)
    {

    }
}
```

ServerRpc:tä käytettäessä itse metodi toteutetaan palvelimen puolella, jolloin asiakasohjelman puolella tapahtuu vain RPC:n kutsuminen. Kutsuminen toimii samalla tavalla kuin kaikkien muidenkin metodien kanssa. ServerRpc:n kutsuminen onnistuu asiakasohjelman lisäksi myös palvelinta ylläpitävältä isäntäasiakasohjelmalta. Kutsu ei juurikaan poikkea pelkän asiakasohjelman ja palvelimen välisestä kutsusta. Erona on vain se, että kutsu ja toteutus tapahtuvat samalla ohjelmalla. (Unity, 2022-b)

3.6.2 ClientRpc

ClientRpc:n kutsuminen tapahtuu palvelimen puolella, jolloin yksi tai useampi asiakasohjelma suorittaa halutut toiminnot. ClientRpc muodostetaan samalla tavalla kuin ServerRpc, mutta metodissa ja attribuutissa käytetään sen sijaan ClientRpc-päätettä. ClientRpc:n käytössä oletus on, että kutsu koskee kaikkia pelissä olevia asiakasohjelmia, jolloin jokainen asiakasohjelma suorittaa kutsuun sisällytetyt toiminnallisuudet. Kutsu voidaan kuitenkin yksilöidä lisäämällä skriptiin ClientRpcSendParams-parametri, johon asetetaan yhden tai useamman halutun asiakasohjelman tunniste (TargetClientIds), joihin haluamme kutsun vaikuttavan. Myös ClientRpc-kutsu voidaan tehdä palvelimen isäntäasiakasohjelmalta. (Unity, 2022-c)

3.7 NetworkVariable

NetworkVariable on toinen vaihtoehto synkronoida tietoa palvelimen ja asiakasohjelmien välillä. Sen sisälle voidaan varastoida arvoja, jotka NetworkVariable.Value-ominaisuudella synkronoidaan asiakasohjelmille. Jotta ominaisuus toimisi, täytyy NetworkVariable määrittää NetworkBehaviour-luokan perivän skriptin omaavassa peliobjektissa. (Unity, 2022-d)

Kun NetworkVariable luodaan, voidaan sen rakentajassa määrittää itse tiedon tyyppi, joka halutaan lähettää, sekä keillä on oikeus lukea ja muuttaa välitettyä tietoa. Oletuksena kaikki pelissä olevat voivat lukea tiedon, tai tieto voidaan näyttää vain peliobjektin omistajalle sekä palvelimelle. Tiedon muuttaminen kuuluu oletuksena vain palvelimelle, mutta rakentajassa voidaan sallia myös peliobjektin omistajalta tulleet muutokset. (Unity, 2022-d)
Ohjelmakoodissa 3 nähdään esimerkkejä siitä, miten NetworkVariable voidaan määrittää.

Ohjelmakoodi 3 NetworkVariable-esimerkkejä.

```
// Esimerkki 1: Lukuoikeudet kaikilla, muutosoikeudet omistajalla
public NetworkVariable<int> TestVariable1 = new NetworkVariable<int> (1,
    NetworkVariableReadPermission.Everyone, NetworkVariableWritePermission.Owner);

// Esimerkki 2: Lukuoikeudet omistajalla, muutosoikeudet palvelimella
public NetworkVariable<bool> TestVariable2 = new NetworkVariable<bool> (false,
    NetworkVariableReadPermission.Owner, NetworkVariableWritePermission.Server);

// Esimerkki 3: Lukuoikeudet kaikilla, muutosoikeudet palvelimella
public NetworkVariable<int> TestVariable3 = new NetworkVariable<int> (2,
    NetworkVariableReadPermission.Everyone, NetworkVariableWritePermission.Server);
```

NetworkVariable ei suoraan tue string-tyyppisiä arvoja. String-arvojen välittämiseen käytetään Unityn omia Fixed String-arvoja, joita on käyttötarpeen mukaan viittä eri kokoa: FixedString32Bytes, FixedString64Bytes, FixedString128Bytes, FixedString512Bytes sekä FixedString4096Bytes. (Unity, 2022-d) Nimessä esiintyvä bittien lukumäärä kertoo, kuinka monta merkkiä arvo voi maksimissaan pitää sisällään. Esimerkiksi FixedString32Bytes-arvo voi pitää sisällään maksimissaan 32 merkkiä. (Code Monkey, 2022)

4 Demon suunnitelma

Tässä työssä toteutettava demo pitää sisällään teoriaosassa läpikäytyjä Netcode for GameObjects-kirjaston moninpelikehityksen mahdollistavia toiminnallisuuksia. Demon tarkoituksena on vaihe vaiheelta käydä läpi, kuinka kirjasto otetaan käyttöön sekä miten pelien toimivuudelle olennaisia toimintoja voidaan kirjaston komponenteilla luoda palvelimella toimiviksi. Demoa itsessään ei voi luokitella peliksi, koska sen avulla vain testiluontoisesti käydään läpi kirjaston toimivuutta. Itse suoranaista pelimekaniikkaa se ei sisällä.

4.1 Toteutustapa

Ennen demon luomista on etukäteen päätetty ja kirjattu ylös moninpelikehitykselle olennaisimpia komponentteja ja toimintoja, joiden toimintaa käydään läpi testausvaiheessa, jossa tarkastellaan miten ne toimivat kahden eri laitteen välillä. Valintoihin päädyttiin lähteitä tutkimalla, jolloin sai selkeän kuvan siitä mitkä kirjaston osat ovat pakollisia toiminnan kannalta ja mitä komponentteja tarvitaan, jotta moninpeleille olennaiset toiminnot voidaan mahdollistaa. Esimerkkeinä NetworkVariable sekä RPC-kutsut, joita ilman tiedolle tapahtuvaa muutosta ei synkronoida asiakasohjelmien välillä. Tällöin pelien tärkeät toiminnot, kuten pisteet tai pelimaailmassa tapahtuvat muutokset eivät päivittyisi muille pelaajille. Kaikkia teoriaosassa mainittuja asioita ei kuitenkaan demossa testata. Esimerkiksi erillisen palvelimen toimintaa ei testata, sillä demoa suunnitellessa tultiin tulokseen, että kahden laitteen välillä toimivan yksinkertaisen demon testaaminen isännöidyllä palvelimella antaa halutut tulokset työtä varten. Demon sisältämät C#-skriptit luodaan Microsoftin Visual Studio -ohjelmalla.

4.2 Toiminnallisuudet

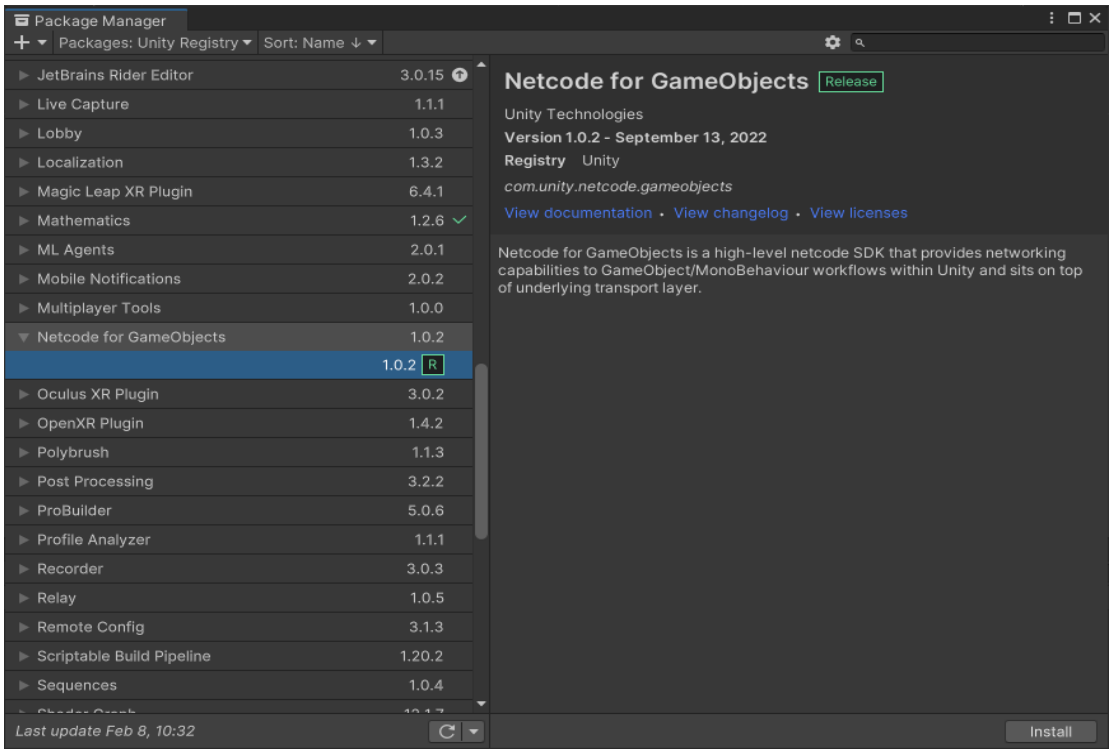
Demo on 2D-projekti, joka pitää sisällään pelaajaobjektin, jonka jokainen asiakasohjelma saa käyttöönsä liittyessään palvelimelle. Pelaajaobjektia pystytään asiakasohjelmalla liikuttamaan ja objektin sijaintiin tehdyt muutokset synkronoidaan näkymään kaikilla muilla palvelimella olevilla asiakasohjelmilla. RPC- sekä NetworkVariable-tiedon synkronointi

toteutetaan luomalla skripti, joka sisältää molempien testaamisen näppäimen painalluksella. Palvelinpuoli toteutetaan lähiverkon avulla. Isäntäpalvelin käynnistetään sille osoitetulla laitteella, jolloin toiset samassa lähiverkossa olevat laitteet pystyvät liittymään palvelimelle.

5 Demon luominen

Tässä työssä kirjaston ominaisuuksia testaava 2D-demo tehdään Unityn versiolla 2021.3.11f1, joka tukee Netcode for GameObjects -kirjastoa. Kirjasto ei oletuksena ole osa Unityä, vaan se asennetaan Package Managerin kautta, joka löytyy Unity editorin yläpalkin Window-valikon alta. Package Manager näyttää ensin projektissa olevat pakkaukset ja Netcode for GameObjects -kirjasto löydetään vaihtamalla hakemistoksi Unity Registry (Kuva 4). Kirjasto ei tule näkyviin, jos Unity-versio ei sitä tue.

Kuva 4 Netcode for GameObjects -kirjasto package managerissa (Unity).



5.1 NetworkManager

Kirjaston toimintaan vaadittavaa välttämätöntä NetworkManager-komponenttia varten luodaan tyhjä peliobjekti, johon komponentti lisätään. Peliobjekti nimetään selkeyden vuoksi komponentin nimellä. Komponentin asetuksista valitaan Select Transport -kohdasta UnityTransport. Tämä lisää NetworkManager-objektiin Unity Transport -protokollakomponentin. Komponenttiin tehdään myöhemmässä vaiheessa palvelimeen ja yhteyteen liittyvät muutokset.

5.2 Pelaajaobjektit

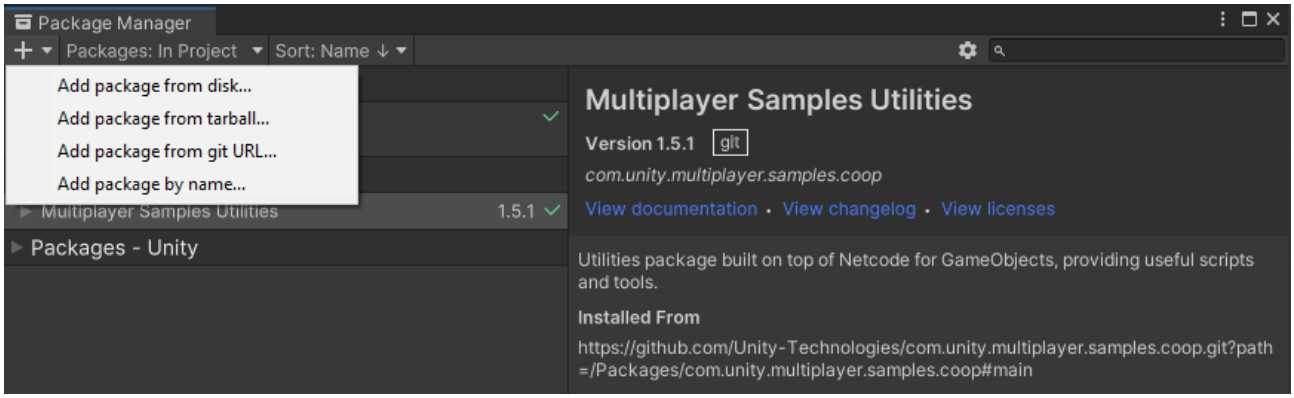
Pelaajan hahmoa esittävänä objektina käytetään demossa yksinkertaista valkoista ympyrää, johon lisätään NetworkObject-komponentti, jotta se saadaan näkymään verkossa. Samalla luodaan projektiin Prefabs-kansio, johon PlayerObject-nimellä luotu pelaajaobjekti lisätään. NetworkManager-komponentti sisältää kohdan Player Prefab, johon lisäämme objektin, jolloin peliin liittyvä pelaaja saa aina kyseisen objektin käyttöönsä. Pelaajaobjekti lisätään myös NetworkPrefabs-listaan, jotta se tunnistettaisiin verkko-objektiksi.

5.2.1 NetworkTransform

Koska demossa halutaan sallia muutosten tekeminen palvelimen ylläpitäjän lisäksi myös asiakasohjelmilta, lisätään ClientNetworkTransform-komponentti pelaajaobjektiin. Komponentti on osa Multiplayer Samples Utilities -pakkausta, jonka asennus tapahtuu Package Managerin kautta liittämällä kuvassa 5 näkyvä URL-osoite kohtaan ”Add package from git URL…”.

Komponentin synkronointiasetuksista jätetään päälle vain sijainnin X-, sekä Y-akselit, sillä muiden tietojen synkronointi on tämän demon toiminnallisuuksien testaamisen kannalta turhaa.

Kuva 5 Multiplayer Samples Utilities -pakkaus (Unity).



5.2.2 Liikkuminen

Pelaajaobjektin liikkumista varten objektiin lisätään komponentit Rigidbody2D, joka mahdollistaa objektin fysiikan sekä Network Rigidbody 2D, joka vaaditaan fysiikan toimimiseen palvelimella. Koska halutaan, että myös asiakasohjelmien muutokset näkyvät, lisätään ClientNetworkTransform-komponentti ennen Network Rigidbody 2D -komponentin lisäämistä, sillä Network Rigidbody 2D:n lisäämisen yhteydessä objektiin lisätään automaattisesti NetworkTransform-komponentti, joka ei tue asiakasohjelmilla tapahtuvia muutoksia. Rigidbody-komponentit tulevat tarpeeseen skriptissä, johon itse objektin liikkuminen määritetään. Skripti (Ohjelmakoodi 4) sisältää tarvittavat muuttujat ja metodit, joilla liikkuminen mahdollistetaan. Start-metodi hakee objektin Rigidbody-komponentin, jota Update-metodi käyttää liikkumisen mahdollistamiseen. Update-metodissa tarkistetaan if-lauseella ”!IsOwner”, että vain objektin omistaja pääsee käyttämään metodia, jolloin muut pelaajat eivät pysty liikuttamaan objektia.

Ohjelmakoodi 4 Pelaajan liikuttamisskripti.

```
using System.Collections;
using System.Collections.Generic;
using Unity.Netcode;
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class PlayerScript : NetworkBehaviour {

    //Movement variables
    [SerializeField]
    private float moveSpeed = 0.5f;
    private Vector2 movement;
    private Rigidbody2D rb;

    Unity Message | 0 references
    private void Start()
    {
        rb = this.GetComponent<Rigidbody2D>();
    }
    Unity Message | 0 references
    private void Update()
    {
        if (!IsOwner) return;

        movement.x = Input.GetAxis("Horizontal");

        movement.y = Input.GetAxis("Vertical");

        rb.MovePosition(rb.position + movement * moveSpeed);
    }

}
```

5.3 Tiedon synkronointi

Tiedon synkronoinnin testaamiseen pelaajaobjektiin lisätään DataSync-skripti (Ohjelmakoodi 5), joka sisältää TestServerRpc- ja TestClientRpc-metodit sekä NetworkVariable-muuttujan alustuksen FixedString128Bytes-tyyppisenä. Muuttujan lukuoikeudet on sallittu kaikille ja kirjoitusoikeudet pelaajaobjektin omistajalle. Metodit sisältävät kaksi lokitiedostoon kirjoittavaa kohtaa. Ensimmäinen kirjoittaa lokitiedostoon, että onko kyseessä ClientRPC- vai ServerRPC-kutsu. Toinen kirjoittaa lokiin NetworkVariable-muuttujan arvon. Kaikkiin skriptin lokitiedostoihin kirjoittaviin kohtiin on myös sisällytetty pelaajaobjektien tunniste, jotta voidaan selkeästi nähdä miltä objektilta tieto on lähtenyt. Testien kutsuminen tapahtuu Update-metodissa, jossa määritellään näppäimen painallus, jolla kutsu tehdään.

Ohjelmakoodi 5 DataSync-skripti.

```
using System.Collections;
using System.Collections.Generic;
using Unity.Collections;
using Unity.Netcode;
using UnityEngine;

public class DataSync : NetworkBehaviour {

    private NetworkVariable<FixedString128Bytes> testString = new
NetworkVariable<FixedString128Bytes>
        ("Testing NetworkVariable. Sent from NetworkObjectId: ",
NetworkVariableReadPermission.Everyone,
NetworkVariableWritePermission.Owner);

    private void Update()
    {
        if (!IsOwner) return;

        if (Input.GetKeyDown(KeyCode.R))
        {
            TestServerRpc();
            TestClientRpc();
        }
    }

    [ServerRpc]
    private void TestServerRpc()
    {
        Debug.Log("Server RPC called using a client. NetworkObjectId:
" + NetworkObjectId);
        Debug.Log($"{testString.Value} {NetworkObjectId}");
    }

    [ClientRpc]
    private void TestClientRpc()
    {
```

```
            Debug.Log("Client RPC called using the host. NetworkObjectId:
" + NetworkObjectId);
            Debug.Log($"{testString.Value} {NetworkObjectId}");
        }

    }
```

5.4 Yhteyden luominen

Isäntäpalvelimen toimintaa varten lisätään NetworkManager-komponentissa valitun Unity Transport -protokollan Address-kenttään isäntäkoneen paikallinen IP-osoite. Näin kerrotaan millä laitteella palvelinta ylläpidetään. Tämä mahdollistaa samassa lähiverkossa olevien käyttäjien liittymisen palvelimelle asiakasohjelmillaan muilta laitteilta. Demoon luodaan käyttöliittymä, joka sisältää napit, joilla pystytään helposti joko käynnistämään isäntäpalvelin tai liittymään palvelimelle asiakasohjelmana. Käyttöliittymä on oma peliobjektinsa, johon lisätään NetworkUI-skripti, joka pitää sisällään nappien toiminnallisuudet.

6 Ominaisuuksien testaaminen

Jotta demoa pystyttäisiin testaamaan, se luodaan ajettavaksi ohjelmaksi (build), jota pystytään käyttämään Unityn ulkopuolella. Build-asetuksista valitaan Player Settings -kohta, jossa asetetaan demolle Company Name sekä Product Name. Nämä ovat tärkeitä tarkasteltavien lokitietojen kannalta, sillä ne löytyvät näille määriteltyjen nimien alla olevasta kansiosta txt-tiedostona. Kun ohjelma on luotu, siitä luodaan kopio testaamisen apuna käytettävälle toiselle tietokoneelle, joka toimii asiakasohjelmana.

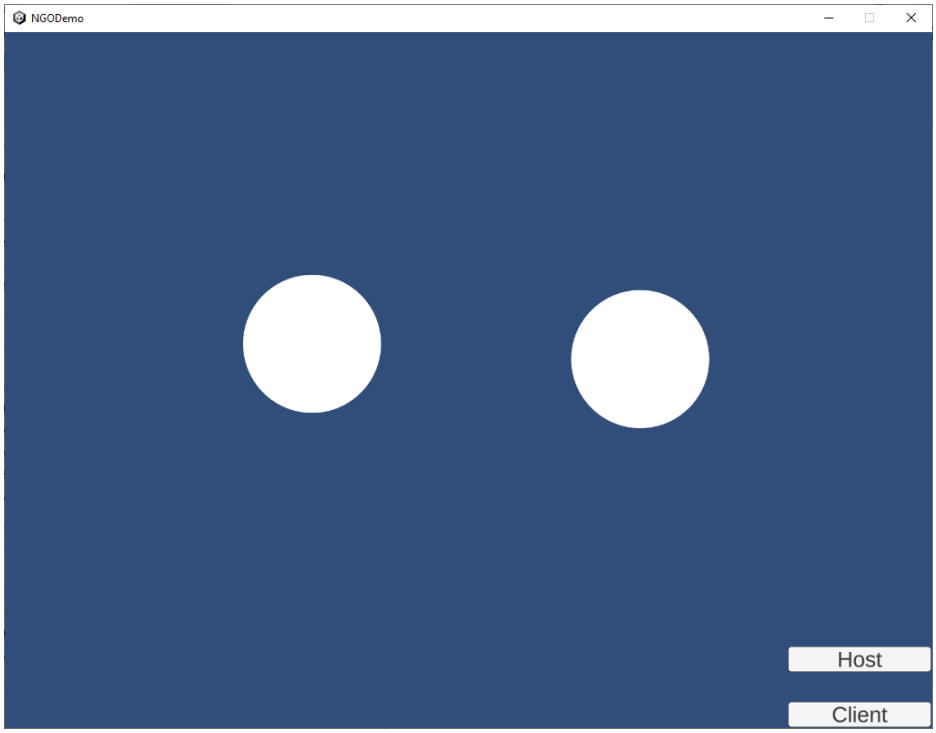
6.1 Yhteys

Yhteyden testaamista varten demo käynnistetään ensin isäntäpalvelimena toimivalla tietokoneella. Palvelin käynnistetään klikkaamalla Host-napista, jolloin isäntäpalvelimena toimiva asiakasohjelma saa itselleen pelaajaobjektin käsiteltäväksi. Toisella tietokoneella käynnissä oleva asiakasohjelma liittyy palvelimelle valitsemalla Client-vaihtoehdon, jolloin sille luodaan uusi pelaajaobjekti käytettäväksi.

6.2 Pelaajaobjektien liikuttaminen

Koska pelaajaobjekteihin on lisätty ClientNetworkTransform-komponentti, näkyy sekä isäntäpalvelinta ylläpitävän asiakasohjelman että toisella tietokoneella olevan asiakasohjelman objektin liikuttaminen palvelimella. Palvelimelle liityttäessä pelaajaobjektit ilmestyvät aina samaan kohtaan, jolloin näyttää siltä, että pelissä olisi vain yksi objekti ennen kuin niitä liikutetaan. Kuvassa 6 nähdään vasemmalla isäntäpalvelimen ja oikealla asiakasohjelman palvelimelle luodut pelaajaobjektit. Objekteja liikuttaessa havaitaan pieni viive asiakasohjelmien välillä.

Kuva 6 Asiakasohjelmien pelaajaobjektit demossa



6.3 Tiedonsiirto

Tiedonsiirron testaamista varten tarvittavat Unityn luomat lokitiedot löydetään oletuksena käyttäjän paikalliselta asemalta AppData-hakemiston LocalLow-kansion sisältä. Kansiosta haetaan projektin Company Name -kansio, joka sisältää lokitiedostot projekteittain. Luodun demon lokitiedot löytyvät tässä tapauksessa Player.log tekstitiedostona kun haetaan C-asemalta ”Users\käyttäjän nimi\AppData\LocalLow\Company Name\Product Name”

Koska isäntäpalvelin on palvelin ja asiakasohjelma samaan aikaan, se lähettää sekä ServerRPC- että ClientRPC-kutsut. Tämä huomataan isäntäpalvelimen lokitiedoissa, joissa esiintyy kaikki kutsut, jotka isäntäpalvelimen asiakasohjelma sekä muut palvelimella olevat asiakasohjelmat tekevät. Tämä nähdään kuvassa 7, jossa juoksevan tunnisteen saaneet asiakasohjelmat ovat lähettäneet RPC-kutsut, jotka sisältävät myös NetworkVariable-muuttujat. Koska isäntäpalvelin käynnistetään ensimmäisenä se saa tunnisteen 1 ja toinen asiakasohjelma tunnisteen 2. Toisella tietokoneella olevan asiakasohjelman lokitiedoista ei nähdä muita tietoja kuin isäntäpalvelimelta tullut ClientRPC-kutsu ja sen sisältö.

Kuva 7 Isäntäpalvelimen lokitietoja.

```
Server RPC called using a client. NetworkObjectId: 1
Testing NetworkVariable. Sent from NetworkObjectId:  1
Client RPC called using the host. NetworkObjectId: 1
Testing NetworkVariable. Sent from NetworkObjectId:  1
Server RPC called using a client. NetworkObjectId: 2
Testing NetworkVariable. Sent from NetworkObjectId:  2
```

7 Tulokset ja pohdinta

Netcode for GameObjects -kirjaston käyttöönotto on yksinkertainen prosessi. Unityn virallinen dokumentaatio ja kirjastosta luodut ohjevideot antavat hyvän pohjan kirjaston ymmärtämiseen. Toiminnallisuuksia testaavan demon luominen onnistui ja se toimi halutulla tavalla. Sitä luodessa kuitenkin törmättiin ongelmiin lokitiedoston lukemisessa, kun NetworkVariable-arvojen käsittely tehtiin erillään RPC-kutsuista. Vähäisellä kokemuksella ei voida varmuudella sanoa oliko vika lokitiedostojen puolella vai oliko skriptissä puutteita. Kirjasto on työn tekovaiheessa vielä melko tuore, joten ongelmien ratkaisussa saattaa esiintyä vaikeuksia, sillä muiden käyttäjien löytämiä ratkaisuja ei löydy niin laajasti. Tämä tulee todennäköisimmin muuttumaan ajan myötä. Koska kirjasto on Unityn virallinen, sen saama tuki ja päivitykset tekevät siitä erittäin hyvän vaihtoehdon moninpelikehitykseen Unityllä.

7.1 Vertailu kolmannen osapuolen kirjaston kanssa

Tämän työn aikana kertyneitä kokemuksia käytiin läpi Akseli Savinaisen kanssa. Akseli teki samanaikaisesti työn kolmannen osapuolen kirjastoista, jossa hän tutki syvemmin Mirror-kirjastoa. Kävimme yhdessä läpi kirjastojen samanlaisuuksia sekä eri toimintoja. Koska molemmat kirjastot perustuvat joillain tavoin Unityn vanhentuneeseen UNet-ratkaisuun, samanlaisuuksia löytyi jonkin verran. Logiikka kirjastojen takana oli melko yhtenäistä ja samanlaisia toiminnallisuuksia löytyi molemmista kirjastoista. Esimerkiksi NetworkManager-komponenttia käytetään molemmilla kehittäessä, mutta ne sisältävät erilaisia asetuksia toisiinsa verrattuna. Myös samalla toimintaperiaatteella toimivia komponentteja löytyi kummastakin, mutta niitä kutsuttiin eri nimillä. Mirror vaikuttaa kirjoitushetkellä olevan aloittelijaystävällisempi ratkaisu, koska se sisältää laajemmin valmiiksi luotuja komponentteja, joita voidaan helposti käyttää toiminnallisuuksien luomiseen ja testaamiseen. Mirror ei vaadi kirjaston itsensä lisäksi ylimääräisiä asennuksia, vaan se sisältää tarvittavat komponentit myös asiakasohjelmalta tulevien muutosten toimintaan, toisin kuin Netcode for GameObjects.

8 Yhteenveto

Tämän opinnäytetyöprosessin aikana löydettiin halutut vastaukset tutkimuskysymyksiin. Kirjaston sisältämiä olennaisia ominaisuuksia ja toimintoja moninpelikehityksen kannalta saatiin selvitettyä. Käyttöönotto ja testaaminen saatiin myös onnistumaan melko vaivatta. Vasta-alkajan näkökulmasta kirjasto pitää sisällään erittäin paljon uutta tietoa, joten ajoittain oli hieman hankalaa ymmärtää varsinaista sekä oikeaa käyttötapaa kaikille toiminnallisuuksille. Tärkeitä osa-alueita jäi vielä laajemman tutkinnan kohteeksi tulevaisuudessa.

Minulla itselläni ei ollut juurikaan mitään kokemusta taikka ymmärrystä moninpelikehityksestä tai palvelimien toiminnasta ennen työn aloittamista. Työtä tehdessäni opin moninpelikehityksen periaatteita sekä eroja eri palvelinmallin valinnan taustalla. Aiemmin kuulemani termistö, jota en ymmärtänyt entuudestaan avautui työtä tehdessä. Opin kattavammin aiheesta, joka minua on kiinnostanut jo pitkään. Videopelien kehityksestä kiinnostuneena koen, että tämän työn kirjoittaminen hyödyttää minua tulevaisuudessa. Esimerkiksi oikean moninpeliprojektin luomiseen koen, että sain hyvän perusymmärryksen ja tietopohjan kirjaston potentiaalista käyttöä varten.

Tämä opinnäytetyö itsessään on vain pintaraapaisu laajaan kokonaisuuteen, mutta sitä voidaan hyödyntää kirjaston perusidean omaksumiseen. Työn kirjoitushetkellä kirjasto on vielä hyvin tuore ja sitä päivitetään sekä parannellaan aktiivisesti, joten aiheella on tulevaisuudessa paljon tutkimus- ja kehityspotentiaalia.

Lähteet

Code Monkey. (26.9.2022). *COMPLETE Unity Multiplayer Tutorial (Netcode for Game Objects*) [video]. YouTube. https://www.youtube.com/watch?v=3yuBOB3VrCk

House, B. (8.9.2020). How to choose the right netcode for your game. *Unity Blog.* https://blog.unity.com/technology/choosing-the-right-netcode-for-your-game

Microsoft. (13.12.2022). *A tour of the C# language.* Microsoft. Haettu 20.1.2023 osoitteesta https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

Mirror. (n.d.). *A Brief History of Mirror.* Mirror. Haettu 23.1.2023 osoitteesta https://mirror-networking.gitbook.io/docs/trivia/a-history-of-mirror

Photon. (n.d.-a). *Introduction.* Photon. Haettu 25.1.2023 osoitteesta https://doc.photonengine.com/en-us/pun/current/getting-started/pun-intro

Photon. (n.d.-b). *Fusion Introduction.* Photon. Haettu 25.1.2023 osoitteesta https://doc.photonengine.com/en-us/fusion/current/getting-started/fusion-intro

Schardon, L. (13.1.2023). *What is Unity? – A guide for one of the top game engines.* GameDev Academy. Haettu 19.1.2023 osoitteesta https://gamedevacademy.org/what-is-unity/

Unity. (30.11.2022-a). *Sending Events with RPCs.* Unity Technologies. Haettu 1.2.2023 osoitteesta https://docs-multiplayer.unity3d.com/netcode/current/advanced-topics/messaging-system

Unity. (25.8.2022-b). *ServerRpc.* Unity Technologies. Haettu 1.2.2023 osoitteesta https://docs-multiplayer.unity3d.com/netcode/current/advanced-topics/message-system/serverrpc

Unity. (17.12.2022-c). *ClientRpc.* Unity Technologies. Haettu 1.2.2023 osoitteesta https://docs-multiplayer.unity3d.com/netcode/current/advanced-topics/message-system/clientrpc

Unity. (8.12.2022-d). *NetworkVariable.* Unity Technologies. Haettu 2.2.2023 osoitteesta https://docs-multiplayer.unity3d.com/netcode/current/basics/networkvariable

Unity. (13.1.2023-a). *Creating and Using Scripts*. Unity Technologies. Haettu 19.1.2023 osoitteesta https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html

Unity. (17.1.2023-b). *Upgrade MLAPI to Netcode for GameObjects.* Unity Technologies. Haettu 25.1.2023 osoitteesta https://docs-

multiplayer.unity3d.com/netcode/current/installation/upgrade_from_mlapi/index.html

Unity. (19.1.2023-c). *About Netcode for GameObjects.* Unity Technologies. Haettu 25.1.2023 osoitteesta https://docs-multiplayer.unity3d.com/netcode/current/about/index.html

Unity. (17.1.2023-d). *NetworkManager.* Unity Technologies. Haettu 27.1.2023 osoitteesta https://docs-multiplayer.unity3d.com/netcode/current/components/networkmanager/index.html

Unity. (25.1.2023-e). *NetworkObject.* Unity Technologies. Haettu 30.1.2023 osoitteesta https://docs-multiplayer.unity3d.com/netcode/current/basics/networkobject

Unity. (17.1.2023-f). *NetworkBehaviour.* Unity Technologies. Haettu 30.1.2023 osoitteesta https://docs-multiplayer.unity3d.com/netcode/current/basics/networkbehavior

Unity. (25.1.2023-g). *NetworkTransform.* Unity Technologies. Haettu 2.2.2023 osoitteesta https://docs-multiplayer.unity3d.com/netcode/current/components/networktransform

Unity. (n.d.-a). *Coding in C# in Unity for Beginners.* Unity Technologies. Haettu 19.1.2023 osoitteesta https://unity.com/how-to/learning-c-sharp-unity-beginners

Unity. (n.d.-b). *Build Multiplayer Games with Netcode for GameObjects.* Unity Technologies. Haettu 25.1.2023 osoitteesta https://unity.com/products/netcode

Unity. (n.d.-c). *An Introduction to Multiplayer Network and Server Models.* Unity Technologies. Haettu 25.1. osoitteesta https://unity.com/how-to/intro-to-network-server-models

Liite 1: Aineistonhallintasuunnitelma

Opinnäytetyön aineisto sisältää itse luotuja kuvakaappauksia sekä ohjelman, joka luodaan työn aikana. Työssä hyödynnetyt lähteistä saadut kuvat ja taulukot merkitään kuvaotsikkoon lähdeviitteellä. Muuten kyseessä on omat kuvakaappaukset.

Työn aineistossa ei ole luottamuksellisia tietoja. Aineisto, joka sisältää kuvat sekä Akseli Savinaisen kanssa käydystä keskustelusta luodun Word-dokumentin säilytetään OneDrive-kansiossa ja siitä luodaan varmuuskopiot tietokoneen paikalliselle asemalle sekä muistitikulle.

Työn aikana luotu demo löytyy sekä paikallisesti tietokoneeltani että GitHub-palvelusta.