



Aleksi Heikkinen

Neljän suora -moninpelin kehitys eri verkkoratkaisulla ja niiden vertailu

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

14.6.2026

Tiivistelmä

Tekijä:	Alexi Heikkinen
Otsikko:	Neljän suora -moninpelin kehitys eri verkkoratkaisulla ja niiden vertailu
Sivumäärä:	25 sivua
Aika:	14.6.2026
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Pelikehitys
Ohjaajat:	Lehtori Antti Laiho

Tämän opinnäytetyön tavoitteena oli vertailla eri verkkoratkaisuja Unity pelimoottorissa käytännön peliprojektin avulla. Työssä toteutettiin Neljän suora -moninpeli käyttäen kolmea eri verkkoratkaisua: Unity Netcode for GameObjectsia, Photon PUN 2:ta ja Alterunaa. Työn tarkoituksena oli tarkastella verkkoratkaisujen käyttöönottoa, käytettävyyttä, kehitysprosessia ja verkkosynkronoinnin toteutusta käytännössä.

Projektin aikana toteutettiin Neljän suora -pelin perustoiminnallisuudet, kuten pelilauta, vuorojärjestelmä, pelinappuloiden mekaniikka ja voittotarkistus. Perusmekaniikkojen valmistuttua peliin lisättiin verkkotoiminnallisuudet jokaiselta verkkoratkaisulta erikseen. Vertailussa tarkasteltiin verkkoratkaisujen ohjelmointitapoja, dokumentaatiota, verkkokutsujen toteutusta ja käyttöönoton helppoutta.

Tulokset osoittivat, että Netcode for GameObjects integroitui selkeimmin pelimoottoriin ja tarjosi suoraviivaisen tavan toteuttaa verkkosynkronointia. Photon tarjosi toimivan RPC-pohjaisen verkkoratkaisun ja valmiit verkkopalvelut moninpelin toteuttamiseen. Alterunan käyttöönotto onnistui verkkoyhteyden osalta, mutta verkkosynkronointi osoittautui haastavammaksi verrattuna muihin toteutuksiin.

Työ osoitti, että eri verkkoratkaisut tarjoavat erilaisia lähestymistapoja moninpelin hallintaan. Soveltuvuus riippuu siten projektin laajuudesta, verkkorakenteesta ja kehittäjän osaamisesta.

Avainsanat: pelinkehitys, moninpeli, verkkoratkaisut, Unity-pelimoottori, verkkosynkronointi

Tämän opinnäytetyön alkuperä on tarkastettu Turnitin Originality Check -ohjelmalla. ChatGPT 5.5 -tekoälymallia on hyödynnetty työn kieliasun ja koodiongelmien tarkistamisessa. Tekijä vastaa itsenäisesti työn lopullisesta sisällöstä.

Abstract

Author:	Aleksi Heikkinen
Title:	Development and comparison of a Connect 4 multiplayer game using different networking solutions
Number of Pages:	25 pages
Date:	14 June 2026
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Game Development
Supervisors:	Antti Laiho, Senior Lecturer

The purpose of this project was to compare different networking solutions in the game engine through a practical game project. A multiplayer Connect 4 game was implemented using three different networking solutions: Unity Netcode for GameObjects, Photon PUN 2 and Alteruna. The goal was to compare the implementation process, usability, networking synchronisation, and overall development workflow of each solution.

During the project, the core gameplay of Connect 4 was developed, including the game board, turn system, piece mechanics, and win detection. Networking features were then implemented separately for each networking solution. The project compared the programming approaches, documentation quality, networking methods, and ease of implementation between the solutions.

Based on the results, Netcode for GameObjects integrated most naturally into the game engine and provided a straightforward approach to network synchronisation. Photon offered a functional RPC-based networking solution together with ready-made online services for multiplayer games. Alteruna successfully established network connections between players, however its synchronisation system proved more challenging to implement compared to other solutions.

The project demonstrated that different networking solutions offer different approaches to multiplayer game management. The suitability therefore depends on the scale of the project, network architecture and the developer's experience

Keywords: game development, multiplayer games, networking solutions, Unity game engine, network synchronisation

The originality of this thesis has been checked using Turnitin Originality Check service. ChatGPT 5.5 was used to check the wording and coding problems of this project. The author remains fully responsible for the final content.

Sisällys

Lyhenteet

1	Johdanto	1
2	Verkkototeutukset	2
2.1	Unity Netcode for GameObjects	2
2.2	Photon	5
2.3	Alteruna	7
3	Verkkototeutuksien vertailu	10
3.1	Käytettävyys ja opittavuus	10
3.2	Suorituskyky	11
3.3	Verkkototeutuksien lisenssit ja kustannukset	12
3.4	Aikaisempia pelejä ratkaisulla	13
3.4.1	Unity Netcode for GameObjects	13
3.4.2	Photon	14
3.4.3	Alteruna	15
4	Pelinkehityksen prosessi	16
4.1	Neljän suora -pelin toteutus	17
4.2	Pelin teko alusta loppuun	18
4.3	Kehitysvaiheen keskeiset havainnot	20
5	Yhteenveto	23
	Lähteet	24

Lyhenteet

- CCU: *Concurrently Connected Users*. Samanaikaisesti palveluun yhdistettyjen käyttäjien määrä.
- FPS: *First Person Shooter*. Videopelien genre, jossa toimintaa seurataan pelihahmon omasta silminnäkijäperspektiivistä.
- LAN: *Local Area Network*. Lähiverkko, jossa laitteet ovat yhteydessä samaan paikalliseen verkkoon.
- PUN: *Photon Unity Networking*. Photonin suunniteltu verkkoratkaisu moninpelien toteuttamiseen Unity-pelimoottorille.
- RPC: *Remote Procedure Call*. Verkkokutsu, jonka avulla funktio voidaan suorittaa toisella laitteella tai palvelimella.
- SDK: *Software Development Kit*. Ohjelmistokehityspaketti, joka sisältää työkaluja ja kirjastoja sovelluskehitystä varten.

1 Johdanto

Videopelien moninpelitoiminnot ovat yleistyneet merkittävästi viime vuosien aikana, ja verkkopelaamisen hallinnasta on tullut keskeinen osa modernia pelinkehitystä. Pelaajat odottavat nykyään mahdollisuutta pelata yhdessä muiden kanssa joko paikallisesti tai internetin välityksellä. Moninpelien toteuttaminen tuo kuitenkin mukanaan useita teknisiä haasteita, kuten tiedonsiirron, pelitilan synkronoinnin ja pelaajien välisen kommunikoinnin hallinnan [1].

Unity-pelimoottorille on saatavilla useita eri vaihtoehtoja verkkopohjaisten pelien toteuttamiseen. Näihin kuuluvat esimerkiksi Unityn oma Netcode for GameObjects -ratkaisu, Photon-verkkokirjasto ja Alteruna-moninpeliratkaisu. Jokainen verkkoratkaisu toimii eri tavoin ja sisältää omat vahvuutensa ja rajoitteensa pelinkehityksen näkökulmasta.

Tämän opinnäytetyön tavoitteena on vertailla eri verkkototeutuksia Unity-pelimoottorissa käytännön peliprojektin avulla. Työssä toteutetaan Neljän suora -moninpeli kolmella eri verkkoratkaisulla, minkä avulla voidaan tarkastella verkkoratkaisujen käyttöönottoa, toimivuutta, kehitysprosessia ja käytettävyyttä. Verkkoratkaisujen vertailu auttaa hahmottamaan, millaiset työkalut soveltuvat parhaiten erilaisten moninpeliprojektien toteuttamiseen Unity-pelimoottorissa.

Neljän suora -peli valittiin projektin toteutettavaksi peliksi sen yksinkertaisen pelimekaniikan ja vuoropohjaisen rakenteen vuoksi. Pelin selkeän logiikan ansiosta työssä voidaan keskittyä verkkototeutusten käyttöönottoon, synkronointiin ja tiedonsiirtoon ilman monimutkaistamalla pelimekaanikkoihin tai grafiikkoihin.

Vertailussa tarkastellaan verkkoratkaisujen käyttöönoton helppoutta, toteutuksen monimutkaisuutta, dokumentaation laatua ja verkkosynkronoinnin toteutusta käytännössä. Työssä ei keskitytä pelin graafiseen toteutukseen tai laajahkojen verkkopalveluiden rakentamiseen.

Opinnäytetyön alussa käsitellään eri verkkototeutuksia ja niiden toimintaperiaatteita Unity-pelimootorissa. Tämän jälkeen työssä esitellään Neljän suora -pelin käytännön toteutus eri verkkoratkaisuilla ja vertaillaan toteutusten ominaisuuksia ja kehitysprosesseja. Lopuksi työssä arvioidaan verkkoratkaisujen soveltuvuutta yksinkertaisen moninpelin toteuttamiseen.

2 Verkkototeutukset

Unity-pelimootorissa moninpelien toteuttamiseen voidaan hyödyntää useita erilaisia verkkoratkaisuja. Yksi ratkaisusta on Unity Technologiesin kehittämä, kun taas loput ovat kolmannen osapuolen tarjoamia verkkokirjastoja ja palveluita. Verkkoratkaisut eroavat toisistaan muun muassa käyttöänoton, arkkitehtuurin, verkkosynkronoinnin ja tarjottujen ominaisuuksien osalta.

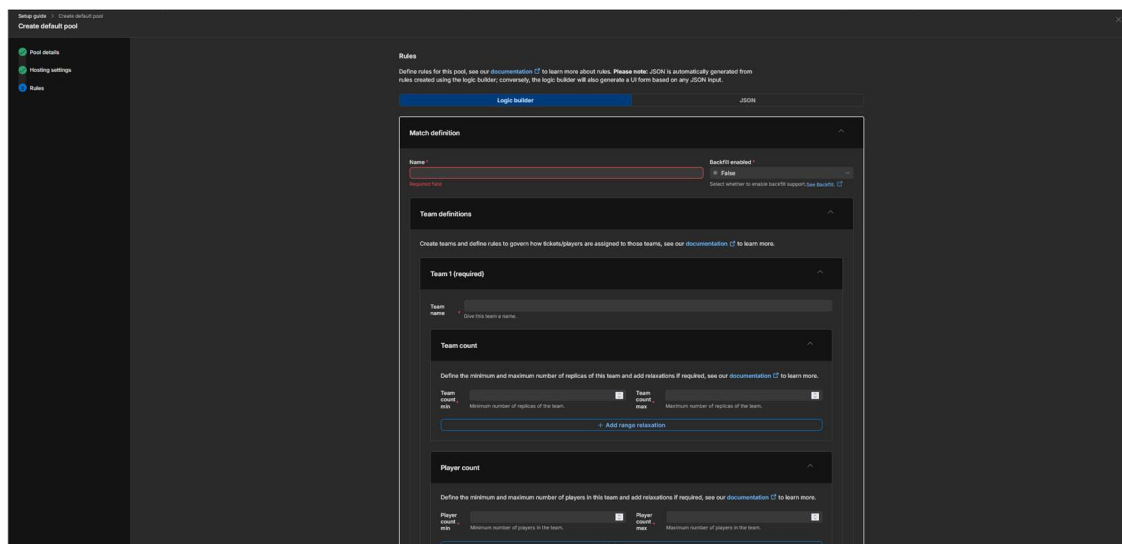
Tässä työssä verkkototeutusten vertailuun valittiin Unity Netcode for GameObjects, Photon PUN 2 ja Alteruna. Valitut verkkoratkaisut edustavat erilaisia lähestymistapoja moninpelien toteuttamiseen Unity-pelimootorissa. Jokainen verkkototeutus toteutettiin saman Neljän suora -pelin pohjalta, jotta verkkoratkaisujen ominaisuuksia ja kehitysprosesseja voitiin vertailla mahdollisimman tasapuolisesti.

2.1 Unity Netcode for GameObjects

Unity Netcode for GameObjects on Unity Technologiesin kehittämä verkkototeutuspaketti Unity-pelimootorille. Paketti mahdollistaa moninpelitoiminnallisuuden toteuttamisen Unity-projekteihin ja pelitilan synkronoinnin pelaajien välillä. [2.]

Unity tarjoaa kattavan dokumentaation ja vaiheittaisia ohjeita verkkototeutusten käyttöönottoon. Ohjeiden avulla kehittäjä pystyy toteuttamaan moninpeliprojektin vaihe vaiheelta samalla oppien verkkototeutuksien toimintaperiaatteita ja ominaisuuksia. [3.]

Unityn verkkototeutuksen kehitys aloitettiin Unity Matchmaker -palvelun avulla. Palvelun tarkoituksena on helpottaa moninpeliyhteyksien hallintaa ja pelaajien yhdistämistä verkkopelissä. Käyttöönoton aikana havaittiin kuitenkin, että palvelu sisälsi useita määrittäsvaiheita ja asetuksia, jotka olivat osittain liian laajoja Neljän suora -projektin yksinkertaiseen kahden pelaajan verkkototeutukseen nähden. Kuva 1 havainnollistaa Unity Matchmaker -palvelun määrittäsnäkymää.



Kuva 1. Unity Matchmaker-palvelun käyttö.

Tämän vuoksi projektissa päätettiin siirtyä käyttämään Unity Netcode for GameObjects -verkkoratkaisua, jonka ratkaisu soveltui paremmin yksinkertaisen vuoropohjaisen moninpelin toteuttamiseen.

Unity Netcode for GameObjects -toteutuksessa projektin verkkotoiminnot rakennettiin isäntä-asiakas-mallin ympärille. Ensimmäinen pelaaja käynnistää pelin isäntänä, jolloin sama sovellus toimii sekä palvelimena että pelaajana. Toinen pelaaja liittyy peliin asiakasroolissa. Yhteyden muodostaminen toteutettiin Unityn NetworkManager- ja UnityTransport-komponenttien avulla.

Neljän suora -pelin verkkosynkronointi toteutettiin lähettämällä pelaajan tekemä siirto pilveen, sen sijaan että koko pelilauta synkronoitaisiin jatkuvasti. Pelaajan valitsema sarake lähetetään palvelimelle RPC-kutsun (Remote Procedure Call,

etäproseduurikutsu) avulla, minkä jälkeen palvelin välittää tiedon molemmille pelaajille. Tämän seurauksena molempien pelaajien pelilaudat päivittyvät samalla tavalla [4]. Esimerkkikoodi 1 havainnollistaa ServerRpc-kutsua, jossa palvelin tarkistaa pelaajan vuoron ennen siirron hyväksymistä.

```
[ServerRpc(RequireOwnership = false)]
void SubmitMoveServerRpc(int column, ServerRpcParams rpcParams = default) {
    if(gameOver) return;
    if(!playersAssigned) return;

    ulong senderId = rpcParams.receive.SenderClientId;
    ulong expectedClient = currentTurnPlayer == 1 ? player1ClientId :
player2ClientId;

    if (senderId != expectedClient)
        return;

    int row = GetLowestEmptyRow(column);

    if (row == -1)
        return;

    PlayMoveClientRpc(column, row, currentTurnPlayer);
}
```

Esimerkkikoodi 1. Palvelimen tekemä vuorotarkistus siirron yhteydessä.

Esimerkkikoodissa asiakas lähettää palvelimelle tiedon valitusta sarakkeesta SubmitMoveServerRpc-kutsun avulla. Palvelin tarkistaa ensin, onko peli päättynyt ja ovatko pelaajat yhdistettynä peliin. Tämän jälkeen palvelin vertailee siirron lähettänyttä pelaajaa nykyiseen pelivuoroon. Jos siirto hyväksytään, palvelin etsii valitusta sarakkeesta alimman vapaan rivin ja välittää siirron molemmille pelaajille PlayMoveClientRpc-kutsun avulla.

Unity Netcode for GameObjects soveltui hyvin vuoropohjaisen moninpelin toteuttamiseen. Verkkototeutuksen käyttöönotto oli johdonmukaista ja suoraviivaista Unityn dokumentaation ja valmiiden ohjeiden ansiosta. Erityisesti RPC-kutsujen käyttö helpotti pelaajien siirtojen synkronointia palvelimen ja asiakkaiden välillä. Toteutuksen aikana havaittiin kuitenkin, että verkkosynkronoinnin hallinta vaatii huolellista vuorojen ja pelaajakohtaisten toimintojen tarkistamista palvelimessa.

2.2 Photon

Photon on kolmannen osapuolen verkkoratkaisu, jota voidaan käyttää moninpeliin toteuttamiseen Unity-pelimoottorissa. Tässä työssä Photon-toteutuksessa käytettiin Photon PUN 2 Free -pakettia, joka tarjoaa valmiita työkaluja verkkoyhteyksien, huoneiden ja pelaajien välisen viestinnän hallintaan. [5.]

Photon eroaa Unity Netcode for GameObjects -ratkaisusta siinä, että verkkoyhteys muodostetaan Photonin tarjoaman palvelimen kautta. Pelaajat yhdistetään huoneisiin, joiden kautta pelin verkkoviestintä tapahtuu. Tässä projektissa ensimmäinen pelaaja luo huoneen ja toinen pelaaja liittyy samaan huoneeseen. Tämän jälkeen pelaajien tekemät siirrot välitetään Photonin RPC-kutsujen avulla molemmille pelaajille. [6.]

Photonin käyttöönotto oli osittain suoraviivaista, sillä aikaisempi kokemus Photonin käytöstä helpotti perustoimintojen ymmärtämistä. Toteutuksen muokkaaminen Unity Netcode for GameObjects -versiosta Photon-versioon vaati kuitenkin tarkkuutta, sillä koodista tuli erottaa, mitkä osat liittyivät pelin yleiseen logiikkaan ja mitkä olivat Unity Netcoden verkkokohtaisia toimintoja.

Photon-toteutuksessa Unity Netcode for GameObjectsin ServerRpc- ja ClientRpc-kutsut korvattiin Photonin omilla RPC-kutsuilla. Verkkoviestintä toteutettiin photonView.RPC()-funktion avulla, jonka kautta pelaajan tekemä siirto välitettiin palvelimelle. Esimerkkikoodi 2 havainnollistaa RPC-kutsun lähettämistä Photon-toteutuksessa.

```
public void RequestDropPiece() {

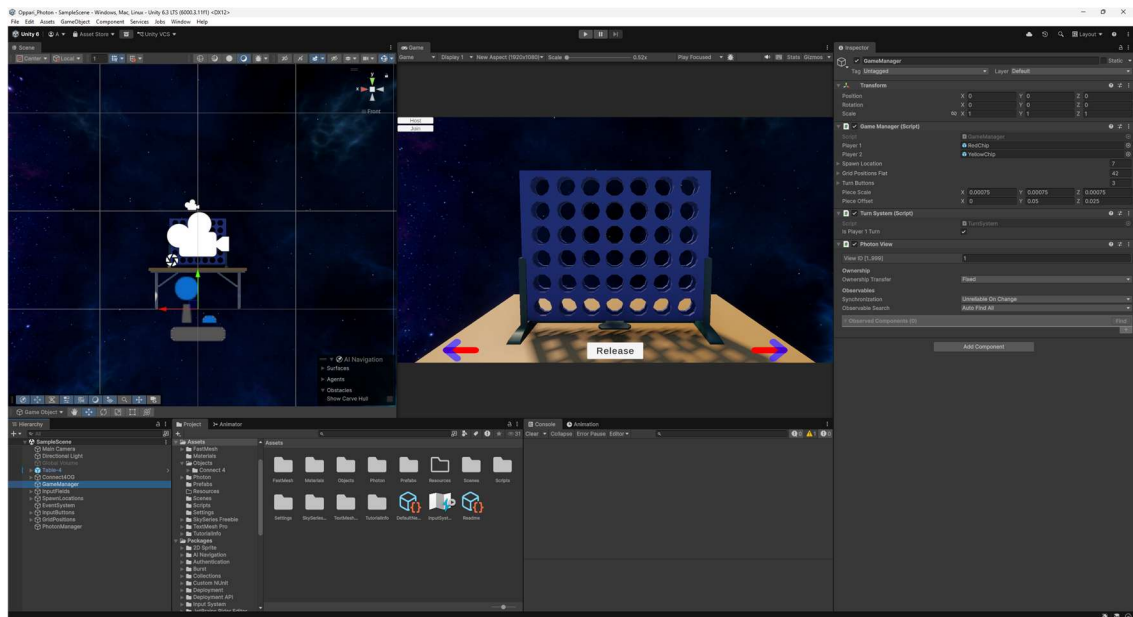
    if (!PhotonNetwork.InRoom) return;
    if (!playersAssigned) return;
    if (!IsMyTurn()) return;

    photonView.RPC(
        "RPC_SubmitMove",
        RpcTarget.MasterClient,
        currentColumn,
        PhotonNetwork.LocalPlayer.ActorNumber
    );
}
```

Esimerkkikoodi 2. Photon RPC -kutsu pelaajan siirron lähettämiseksi palvelimelle.

Esimerkkikoodissa pelaajan tekemä siirto lähetetään Photonin palvelimelle photonView.RPC()-funktion avulla. RpcTarget.MasterClient määrittää, että viesti lähetetään huoneen pääasiakkaalle, joka toimii pelin palvelimen kaltaisessa roolissa. Tämän jälkeen palvelin tarkistaa siirron ja välittää sen edelleen muille pelaajille.

Photon-toteutuksessa keskeinen komponentti on PhotonView. PhotonView mahdollistaa Photonin RPC-kutsujen käytön ja yhdistää verkkoviestit oikeaan peliobjektiin [7]. Ilman PhotonView-komponenttia GameManager-luokan RPC-kutsut eivät toimi, koska Photon ei pysty tunnistamaan, minkä objektin kautta verkkoviestintä tapahtuu. Kuva 2 havainnollistaa PhotonView-komponentin käyttöä GameManager-objektissa Photon-toteutuksessa.



Kuva 2. PhotonView-komponentti GameManager-objektissa.

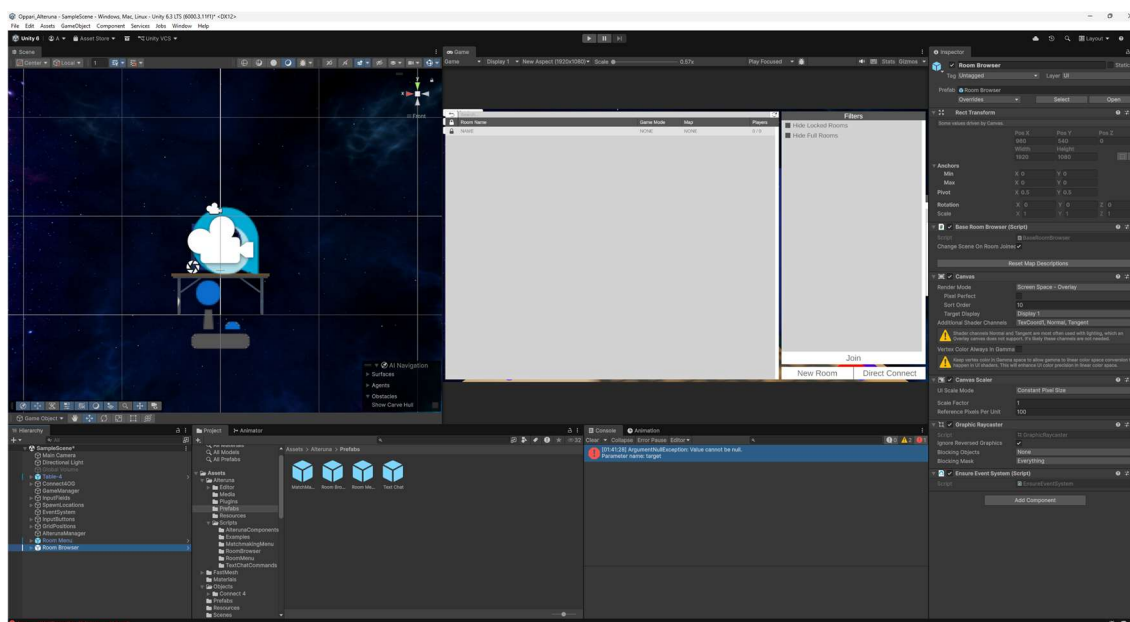
PhotonView-komponentin avulla Photon pystyy yhdistämään RPC-kutsut oikeaan peliobjektiin. Tämä oli tärkeää erityisesti GameManager-luokassa, koska pelin siirtojen lähettäminen ja vastaanottaminen toteutettiin saman objektin kautta. Photon PUN 2 Free -versiossa on myös rajoituksia, joista yksi liittyy CCU-määrään. CCU tarkoittaa samanaikaisesti yhdistettyjä käyttäjiä eli sitä, kuinka monta käyttäjää voi olla yhteydessä palveluun yhtä aikaa. Ilmaisversiossa samanaikaisten käyttäjien määrä on rajattu 20 käyttäjään [8]. Neljän suora-projektin kaltaisessa kahden pelaajan prototyypissä käyttäjämäärä ei aiheuttanut ongelmia, mutta laajemmissa monipeliprojekteissa rajoitukset tulisi huomioida jo suunnitteluvaiheessa.

2.3 Alteruna

Alteruna on kolmannen osapuolen verkkototeutus, jota käytettiin työn kolmantena vertailevana verkkoratkaisuna. Alteruna tarjoaa Unity-pelimoottorille suunnatun monipeliratkaisun, joka sisältää valmiita työkaluja verkkoyhteyksien, huoneiden ja synkronoitavien objektien hallintaan. Alteruna tarjoaa myös valmiita valmismalli- (prefab) ja komponenttiratkaisuja verkkotoiminnallisuuksien toteuttamiseen [9].

Alteruna poikkesi rakenteeltaan selvästi Unity Netcode for GameObjects- ja Photon-ratkaisuista. Verkkotoiminnallisuuksien toteutus perustui enemmän Alterunan omiin prefab- ja komponenttiratkaisuihin kuin suoraan ohjelmallisiin verkkokutsuihin. Lisäksi havaittiin, että Unity Asset Store -version ja Alterunan omilta verkkosivuilta ladattavan version välillä oli eroja käyttöönotossa ja toiminnallisuuksissa.

Alteruna tarjoaa valmiita työkaluja huoneiden ja verkkoyhteyksien hallintaan. Kuvassa 3 näkyy Alterunan valmis huonejärjestelmä, jonka avulla pelaajat voivat luoda ja selata aktiivista huoneita.



Kuva 3. Alterunan valmis huonejärjestelmä.

Projektissa valmista huonejärjestelmää ei kuitenkaan hyödynnetty lopullisessa toteutuksessa, koska sen käyttöönotossa ilmeni ongelmia eikä huoneisiin yhdistäminen toiminut projektin aikana luotettavasti. Tämän vuoksi verkkoyhteys toteutettiin Alterunan dokumentaation mukaisella yksinkertaisemmalla host- ja join-ratkaisulla [9].

Alterunan-version toteutuksessa yhteys isäntänä ja asiakkaan välille saatiin muodostettua onnistuneesti. Pelaajat pystyivät liittymään samaan

verkkosessioon, ja yhteyden muodostaminen näkyi myös Unity-editorin konsolissa. Pelin siirtojen verkkosynkronointi osoittautui kuitenkin selvästi haastavammaksi kuin Unity Netcode for GameObjects- ja Photon-toteutuksissa.

Erityisesti verkkokutsujen ja synkronoitavien objektien hallinta vaati enemmän perehtymistä Alterunan omaan järjestelmään. Alterunan verkkototeutus perustui Synchronizable-objekteihin ja niiden yksilöllisiin tunnisteesiin (Unique ID), joiden avulla objektit tunnistetaan palvelimen kautta [10].

Lisäksi verkkokutsujen toteuttaminen poikkesi huomattavasti aiemmin käytetyistä RPC-ratkaisuista. Esimerkkikoodi 3 havainnollistaa pelaajan siirron lähettämistä Alteruna-version verkkototeutuksessa.

```
public void RequestDropPiece() {
    if (gameOver) return;
    if (moveSender != null) {
        moveSender.SendMove(currentColun, currentTurnPlayer);
    }
}
```

Esimerkkikoodi 3. Alteruna-version siirron lähettäminen verkkoyhteyden kautta.

Esimerkkikoodissa pelaajan tekemä siirto välitetään AlterunaMoveSender-luokan kautta toiselle pelaajalle. Ennen siirron lähettämistä tarkistetaan, ettei peli ole päättynyt ja että verkkoyhteydestä vastaava moveSender-objekti on olemassa. Toteutus poikkesi Unity Netcode for GameObjects- ja Photon-versioista, jossa verkkokutsut toteutettiin suoraan RPC-kutsujen avulla.

Projektin aikana verkkoyhteyden muodostaminen onnistui, mutta Neljän suora -pelin siirtojen täydellistä synkronointia ei saatu toteutettua samalla tasolla kuin Unity Netcode for GameObjects- ja Photon-versioissa. Tämä osoitti käytännössä, että Alterunan käyttöönotto vaati enemmän perehtymistä sen omaan verkkorakenteeseen ja valmiisiin verkkokomponentteihin.

3 Verkkototeutuksien vertailu

3.1 Käytettävyys ja opittavuus

Verkkoratkaisujen välillä havaittiin selkeitä eroja niiden käyttöönotossa, ohjelmointitavassa ja opittavuudessa. Unity Netcode for GameObjects oli toteutuksesta selvästi eniten koodipainoinen ratkaisu. Verkkotoiminnallisuudet rakennettiin pääosin ohjelmallisesti hyödyntämällä ServerRpc-, ClientRpc-kutsuja ja Unityn omia verkkokomponentteja. Koodipainotteisuus mahdollisti verkkoliikenteen ja pelitilan tarkan ohjelmallisen kontrollin, mutta samalla vaati enemmän ymmärrystä verkkosynkronoinnista ja palvelin-asiakasrakenteesta.

Photon puolestaan yhdisti ohjelmallisen verkkototeutuksen ja verkkopalveluiden hallinnan. Photonin käyttöönotossa keskeisessä roolissa oli Photon Dashboard jaa AppId-tunnuksen käyttö, joiden avulla projekti yhdistettiin Photonin palveluihin. Photonin RPC-järjestelmä muistutti toimintaperiaatteeltaan Unity Netcode for GameObjects -ratkaisua, minkä vuoksi siirtyminen Netcode for GameObjects-toteutuksesta Photon-versioon oli suhteellisen suoraviivaista ohjelmoijan Photon-kokemuksen ansiosta.

Alteruna erosi kahdesta muusta verkkoratkaisusta selvästi. Toteutus painottui enemmän valmiisiin prefab-ratkaisuihin, käyttöliittymäkomponentteihin ja Alterunan omiin verkkokomponentteihin kuin suoraan ohjelmallisiin RPC-kutsuihin. Tämä helpotti joidenkin verkkotoimintojen käyttöönottoa, mutta samalla verkkorakenteen ymmärtäminen osoittautui vaikeammaksi. Erityisesti synkronoitavien objektien ja verkkokutsujen toiminta poikkesi merkittävästi Unity Netcode for GameObjects- ja Photon-ratkaisuista.

Dokumentaation laadusta havaittiin myös eroja verkkoratkaisujen välillä. Unity Netcode for GameObjects tarjosi kattavan dokumentaation ja vaiheittaisia opeusmateriaaleja, jotka helpottivat verkkototeutuksen oppimista. Photonin dokumentaatio keskittyi enemmän valmiisiin verkkoratkaisuihin ja palveluiden käyttöönottoon. Alterunan dokumentaatio puolestaan sisälsi useita valmiita

esimerkkiprojekteja, mutta joidenkin ominaisuuksien käyttöönotto jäi osittain epäselväksi projektin aikana.

Kokonaisuudessaan Unity Netcode for GameObjects oli verkkototeutuksista selkein oppia Unity-pelimoottorin näkökulmasta, kun taas Photon tarjosi tasapainoisen yhdistelmän ohjelmallista hallintaa ja valmiita verkkopalveluita. Alteruna puolestaan vaati enemmän perehtymistä sen omiin verkkorakenteisiin ja toimintatapoihin ennen kuin verkkosynkronointia saatiin toteutettua toimivasti.

3.2 Suorituskyky

Projektin aikana verkkototeutusten suorituskykyä tarkasteltiin käytännön testauksen kautta. Neljän suora -pelin kaltaisessa vuoropohjaisessa moninpelissä verkkoliikenteen määrä pysyi suhteellisen pienenä, koska palvelimeen lähetettiin ainoastaan pelaajien tekemät siirrot eikä koko pelitilaa synkronoitu jatkuvasti.

Unity Netcode for GameObjects -toteutuksessa verkkosynkronointi toimi vakaasti ja ennakoitavasti. ServerRpc- ja ClientRpc-kutsujen avulla pelaajien siirrot välittyivät nopeasti molemmille osapuolille ilman havaittavaa viivettä. Koska toteutus hyödynsi Unityn omia verkkokomponentteja, verkkotoiminnallisuudet integroituvat hyvin osaksi muuta projektia.

Photon-toteutuksessa suorituskyky vastasi pitkälti Unity Netcode for GameObjects -toteutusta. Pelaajien siirrot välittyivät nopeasti Photonin RPC-kutsujen avulla, eikä projektin aikana havaittu merkittäviä viiveitä tai synkronointiongelmia kahden pelaajan välillä. Photonin palvelinpohjainen verkkoratkaisu mahdollisti myös verkkoyhteyden hallinnan ilman erillistä palvelinlogiikan toteutusta projektin sisällä.

Alteruna-toteutuksessa verkkoyhteyden muodostaminen onnistui, mutta verkkosynkronointi ei toiminut yhtä vakaasti kuin kahdessa muussa verkkoratkaisussa. Pelaajien välinen yhteys muodostui onnistuneesti host- ja join-ratkaisun

avulla, mutta pelin siirtojen synkronoinnissa esiintyi ongelmia. Tämän vuoksi Alterunan suorituskykyä ei voitu arvioida täysin samalla tasolla kuin Unity Netcode for GameObjects- ja Photon-toteutuksia.

Projektin perusteella Unity Netcode for GameObjects ja Photon soveltuivat hyvin Neljän suora -pelin kaltaiseen kevyeen vuoropohjaiseen moninpeliin. Molemmissa ratkaisuisa verkkoliikenteen määrä pysyi pienenä, koska palvelimen kautta välitettiin vain pelaajien tekemät siirrot. Tämä vähensi tarpeettoman verkkoliikenteen määrää ja yksinkertaisti verkkosynkronoinnin toteutusta.

3.3 Verkkototeutuksien lisenssit ja kustannukset

Kaikista tässä työssä käytetyistä verkkoratkaisuista oli saatavilla ilmaiset versiot, joita hyödynnettiin projektin toteutuksessa. Verkkoratkaisujen käyttöönotto oli suhteellisen nopeaa eikä niiden käyttö vaatinut merkittäviä alkuinvestointeja. Tämä mahdollisti moninpelitoiminnallisuuksien toteuttamisen maksuttomasti kehitysvaiheen aikana.

Unity Netcode for GameObjects sisältyy Unity-pelimoottorin ekosysteemiin, ja sen käyttöönotto tapahtuu Unityn Package Manager -järjestelmän kautta. Verkkoratkaisun käyttö ei vaadi erillisiä lisenssimaksuja, mikä tekee siitä kustannustehokkaan vaihtoehdon pienten moninpeliprojektien toteuttamiseen.

Photon PUN 2 tarjoaa ilmaisen version, jossa samanaikaisesti yhdistettyjen käyttäjien määrä on rajoitettu. Ilmainen versio soveltui hyvin tämän projektin kaltaiseen kahden pelaajan moninpelitoteutukseen. Photon tarjoaa lisäksi maksullisia versioita, joiden avulla palvelimen käyttäjämäärää voidaan kasvattaa laajoja moninpeliprojekteja varten.

Alterunan verkkototeutuksen käyttöönotto toteutettiin aluksi Unity Asset Storesta ladattavan maksuttoman version avulla. Ohjelmisto tarjoaa useita eri lisenssi ja käyttösuunnitelmia eri käyttäjäryhmille, kuten indie-kehittäjille, studioille, oppilaitoksille ja yrityksille. Ilmainen versio tarjoaa valmiit työkalut

pienimuotoisten LAN-pohjaisten moninpelien toteuttamiseen, mikä soveltui hyvin tämän projektin kahden pelaajan Neljän suora -toteutukseen.

Maksulliset versiot mahdollistavat laajat verkkopalvelut ja suuren palvelinkapasiteetin. Tämän vuoksi Alteruna soveltuu myös suurempien moninpeliprojektien toteuttamiseen riippuen projektien tarpeista ja verkkoinfrastruktuurin laajuudesta.

Projektin aikana havaittiin kuitenkin ongelmia Unity Asset Store -version toiminnassa minkä vuoksi verkkototeutuksen käyttöönotto siirrettiin Alterunan omilta verkkosivuilta ja GitHubista ladattavaan versioon [11]. GitHub-version dokumentaatioiden ja ohjeiden avulla verkkoyhteyden muodostaminen saatiin toteutettua toimivasti.

Verkkoratkaisujen lisäksi Unity-projekteissa voidaan hyödyntää erilaisia lisäosia ja Asset Store -sisältöjä, joihin saattaa liittyä omia lisenssiehtoja ja -kustannuksia. Tässä projektissa käytettiin ilmaista 3D-mallia pelilaudan toteuttamiseen, minkä vuoksi myös ulkopuolisten materiaalien lisenssit tuli huomioida projektin toteutuksessa.

3.4 Aikaisempia pelejä ratkaisulla

Verkkoratkaisujen vertailun tueksi oli hyödyllistä tarkastella myös aikaisempia projekteja ja pelejä, joissa samoja verkkoteknologioita on hyödynnetty. Kaikkien pelien osalta ei kuitenkaan ole julkisesti saatavilla tarkkaa tietoa käytetystä verkkoratkaisun versiosta. Tämän vuoksi raportin esimerkit perustuvat joko verkkoratkaisujen virallisiin esimerkkiprojekteihin tai julkisesti saatavilla oleviin tietoihin verkkoteknologioiden käytöstä.

3.4.1 Unity Netcode for GameObjects

Unity tarjoaa Netcode for GameObjects -verkkoratkaisun tueksi useita virallisia esimerkkiprojekteja. Näistä tunnetuimpia ovat Galactic Kittens ja Boss Room

[12; 13]. Molemmat projektit on toteutettu Unityn omilla verkkotyökaluilla, ja niiden tarkoituksena on havainnollistaa verkkosynkronoinnin ja moninpelitoimintojen toteutusta käytännössä.

Unity on julkaissut myös artikkelin Netcode for GameObjects -verkkoratkaisun käyttöönotosta ja moninpelin kehitysprosessista [14]. Artikkelissa käsitellään muun muassa verkkoyhteyksien muodostamista, RPC-kutsujen hyödyntämistä ja pelitilan synkronointia pelaajien välillä. Samoja periaatteita hyödynnettiin myös työn Neljän suora -projektissa.

Neljän suora -projekti oli rakenteeltaan huomattavasti yksinkertaisempi kuin Galactic Kittens tai Boss Room. Palvelimen kautta välitettiin pääasiassa yksittäisiä siirtotapahtumia, kun taas esimerkkiprojektissa synkronoidaan jatkuvasti pelaajien liikkeitä, animaatioita ja pelimaailman tapahtumia. Tästä huolimatta verkko-toteutuksen peruseriaatteen olivat samankaltaisia, sillä kaikissa projekteissa hyödynnettiin RPC-kutsuja ja palvelimen kautta tapahtuvia pelitilan synkronointia.

3.4.2 Photon

Photon-teknologiaa on käytetty useissa tunnetuissa moninpeleissä. Esimerkkinä ovat Phasmophobia [15], joka hyödyntää Photonin verkkoratkaisuja pelaajien välisen verkkoviestinnän toteuttamiseen. Toteutettu projekti osoittaa, että Photon-teknologia soveltuu sekä pieniin yhteistyöpeleihin että isoihin moninpeleihin. Photonin julkaisemien tuotetietojen ja asiakasreferenssien mukaan teknologiaa käytetään useissa kaupallisissa moninpeleissä eri alustoilla [16].

Muita Photon PUN 2 -verkkoratkaisua hyödyntäviä projekteja ovat esimerkiksi tekijän aikaisempi kokemus Ah, Rats! -peliprojektin parissa. Projekti toteutettiin osana ammattikorkeakoulun projektikurssia. Kyseinen peli ei tällä hetkellä ole julkisesti saatavilla. Pelissä 2–8 pelaajaa pelaa joko rottina tai ihmisenä. Rottien tavoitteena on suorittaa erilaisia tehtäviä ja lisätä ihmispelaajan stressitasoa, kun taas ihmisen tavoitteena on estää rottien toiminta ja kukistaa ne.

Projektin toteutuksessa Photon PUN 2 mahdollisti pelaajien yhdistämisen samaan pelisessioon ja pelitilaan synkronoinnin. Projektiryhmä oli kokonaisuudessaan tyytyväinen verkkoratkaisun toimintaan, ja peli saatiin toteutettua onnistuneesti projektin tavoitteiden mukaisesti.

Projektin aikana havaittiin kuitenkin, että pitkissä pelisessioissa pelitilan synkronointi saattoi ajan myötä heikentyä, mikä johti desynkronoitumiseen pelaajien välillä. Neljän suora -projektissa vastaavaa ongelmaa ei havaittu, koska Neljän suora -pelisessiot ovat huomattavasti lyhyempiä ja palvelimen kautta siirretään vain yksittäisiä siirtotapahtumia. Tämän vuoksi verkkoliikenteen määrä pysyi vähäisenä ja synkronoinnin hallinta oli yksinkertaisempaa.

Toinen merkittävä ero projektien välillä liittyy huonejärjestelmään. Ah, Rats! -projektissa pelaajat pystyivät luomaan useita pelihuoneita ja liittymään niihin, kun taas Neljän suora -projektissa verkkoyhteys toteutettiin yksinkertaisemmalla mallilla, jossa pelaaja joko luo pelin tai liittyy olemassa olevaan peliin. Tämä aiempi kehityskokemus helpotti merkittävästi PhotonView-komponenttien, RPC-kutsujen ja huonejärjestelmän käyttöönottoa Neljän suora -projektissa.

Photon tarjoaa myös useita virallisia esimerkkiprojekteja, joiden avulla kehittäjät voivat tutustua erilaisiin moninpeliratkaisuihin ja verkkosynkronoinnin toteutukseen [16]. Esimerkkiprojektit havainnollistavat muun muassa huonejärjestelmiä, pelaajien synkronointia ja verkkopohjaista pelilogiikkaa.

3.4.3 Alteruna

Alterunan osalta tunnettuja kaupallisia peliesimerkkejä oli huomattavasti vaikeampi löytää kuin Unity Netcode for GameObjects- ja Photon-ratkaisujen tapauksessa. Alteruna esittelee kuitenkin omalla Showcase-sivuillaan useita verkkoratkaisua hyödyntäviä esimerkkiprojekteja, kuten FPS-, Racing-, XR- ja 2D Platformer -projektit [17].

Näiden projektien tarkoituksena on havainnollistaa Alterunan tarjoamia verkkotoimintoja ja tarjota kehittäjille valmiita lähtökohtia omien moninpelien toteuttamiseen. Erityisesti FPS- ja Racing-projektit hyödyntävät jatkuvaa pelaajien sijaintien ja pelitilan synkronointia, mikä asettaa verkkoratkaisulle huomattavasti suuremmat vaatimukset kuin tämän työn Neljän suora -projekti [18; 19].

Neljän suora -projektin toteutus oli huomattavasti yksinkertaisempi, sillä palvelimen kautta siirrettiin pääasiassa yksittäisiä siirtotapahtumia eikä jatkuvaa pelimaailman tilan päivitystä. Tästä huolimatta Alterunan käyttöönotto osoittautui haastavammaksi kuin Unity Netcode for GameObjects- ja Photon-ratkaisuissa. Vaikka verkkoyhteys saatiin muodostettua onnistuneesti, pelitilan synkronointi ei toiminut yhtä luotettavasti kuin muissa toteutuksissa.

Alterunan esimerkkiprojektit osoittivat kuitenkin, että verkkoratkaisulla on mahdollista toteuttaa myös huomattavasti monimutkaisempia moninpelejä kuin tässä työssä toteutettu Neljän suora [18.].

4 Pelinkehityksen prosessi

Tämän insinööritoiminnan projektin käytännön osuus aloitettiin rakentamalla Neljän suora -pelin perustoiminnallisuudet Unity-pelimoottorissa. Projektin tavoitteena oli toteuttaa sama moninpeli kolmella eri verkkoratkaisulla, jotta niiden käyttöönottoa, toimivuutta ja kehitysprosessia voitiin vertailla käytännössä.

Pelin toteutus eteni vaiheittain yksinpelimekaniikoista verkkotoiminnallisuuksiin. Kehityksen aikana projektiin toteutettiin muun muassa pelilaudan logiikka, vuorojärjestelmä, pelinappuloiden pudotusmekaniikka ja voittotarkistus. Tämän jälkeen peliin lisättiin verkkotoiminnallisuudet Unity Netcode for GameObjects-, Photon- ja Alteruna-verkkoratkaisujen avulla.

4.1 Neljän suora -pelin toteutus

Neljän suora -pelin toteutus aloitettiin rakentamalla pelilaudan rakenne ja pelinappuloiden sijoitusjärjestelmä. Pelilaudassa käytettiin seitsemää saraketta ja kuutta riviä, joiden avulla pelaajien nappulat voitiin sijoittaa oikeisiin pelikohtiin. Jokaiselle pelilaudan ruudulle määritettiin oma sijaintinsa Unity-editorissa, minkä avulla nappulat voitiin asettaa oikeille riveille ja sarakkeille.

Pelaajan ohjaama nappula liikkuu pelilaudan yläpuolella vaakasuunnassa painikkeiden avulla. Pelaaja pystyy siirtämään nappulaa vasemmalle tai oikealle ja pudottamaan nappulan haluttuun sarakkeeseen. Pudotuksen jälkeen järjestelmä tarkistaa alimman vapaan rivin ja sijoittaa nappulan oikeaan kohtaan pelilaudalla.

Neljän suora -peli valittiin projektiksi sen yksinkertaisen toteutuksen ja tunnettuuden vuoksi. Pelin logiikka on rakenteeltaan yksinkertainen, mutta pelin toteuttaminen Unity-pelimoottorissa sisälsi useita mielenkiintoisia teknisiä haasteita. Vaikka peli on suunniteltu kahdelle pelaajalle, vuoropohjainen logiikka soveltui hyvin tämän projektin toteutukseen. Pelin yksinkertainen rakenne mahdollisti keskittymisen erityisesti verkkototeutuksiin ja niiden toiminnan tarkasteluun itse pelimekaniikan sijaan.

Pelilaudan toteutuksessa hyödynnettiin ilmaista 3D-mallia pelilaudasta ja nappuloista [20]. Valmiiden mallien käyttö nopeutti projektin kehitystä ja mahdollisti keskittymisen ensisijaisesti verkkotoiminnallisuuksien toteuttamiseen. Vaihtoehtona olisi ollut toteuttaa peli kokonaan itse tehdyllä 3D-mallilla tai 2D-versiona, mutta niiden arvioitiin vievän tarpeettoman paljon aikaa projektin tavoitteisiin nähden.

Pelilaudan rivit ja sarakkeet toteutettiin collider-pohjaisella järjestelmällä. Collider on Unity-pelimoottorissa objektin alueen tunnistava komponentti, jonka avulla voidaan havaita objektien välisiä osumia ja sijainteja. Jokaiselle pelilaudan ruudulle määritettiin oma collider-komponentti ja tieto, mihin sarakkeeseen ruutu kuuluu, minkä avulla järjestelmä pystyy tunnistamaan nappulan sijainnin

ja tarkistamaan, onko kyseinen peliruutu vapaa. Tätä logiikkaa hyödynnettiin nappuloiden sijoittamisessa, voittotilanteiden tarkistuksessa ja pelilaudan tilan hallinnassa.

4.2 Pelin teko alusta loppuun

Projektin toteutus aloitettiin rakentamalla pelialue ja pelaajien nappulat. Molempien pelaajien nappuloista luotiin prefab-objektit, jotka liitettiin GameManager-objektiin. Prefab-objektien avulla peli pystyi luomaan oikean pelaajan nappulan vuoron perusteella ja asettamaan sille oikean koon, sijainnin ja suunnan.

Skriptauksen alkuvaiheessa tavoitteena oli toteuttaa pelistä toimiva paikallinen versio, jossa nappulaa pystyi liikuttamaan ja pudottamaan oikeaan kohtaan pelilaudalla. Kun perustoiminnallisuudet saatiin toimimaan, projektiin lisättiin vuorojärjestelmä ja tarkistukset aiemmin asetettujen nappuloiden tunnistamiseen. Tämän vuorotarkistuksen avulla estettiin nappuloiden päällekkäisyys saman sarakkeen sisällä.

Paikallisen version toteutus oli tärkeä vaihe ennen verkkoratkaisuihin siirtymistä, koska sen avulla pelin peruslogiikka saatiin toimimaan ilman verkkosynkronoinnin tuomia lisähaasteita. Kun nappuloiden liikkuminen, pudotusmekaniikka, vuorojen vaihtuminen ja voittotarkistus toimivat paikallisesti, verkkoratkaisujen lisääminen oli helpompaa toteuttaa vaiheittain. Tämä vähensi myös virheiden määrää, sillä ongelmat voitiin erottaa selkeästi joko pelilogiikkaan tai verkkotoimintoihin liittyviksi.

Verkkoversioiden toteutuksessa pyrittiin säilyttämään pelin perusrakenne mahdollisimman samanlaisena jokaisessa versiossa. Tämän vuoksi GameManager-luokka vastasi edelleen pelilaudan tilasta, vuorojen hallinnasta ja siirtojen tarkistamisesta. Verkkoratkaisujen välillä suurimmat muutokset liittyivät siihen, miten pelaajan tekemä siirto lähetettiin toiselle pelaajalle ja miten siirto hyväksyttiin ennen sen suorittamista.

Paikallisen version valmistumisen jälkeen projektiin toteutettiin verkkotoiminnallisuuksia Unity Netcode for GameObjects -verkkoratkaisun avulla. Toteutuksessa merkittävin vaihe oli pelin toimintojen muuttaminen verkkosynkronoiduksi RPC-kutsujen avulla. Pelaajan tekemä siirto lähetettiin palvelimelle, jonka jälkeen palvelin välitti päivityksen takaisin peliin ja sijoitti nappulan oikeaan kohtaan pelilaudalla.

Projektissa ei toteutettu versiota, jossa pelilaudan yläpuolella liikkuva nappula näkyisi reaaliaikaisesti molemmille pelaajille. Tällaisen ominaisuuden toteuttaminen olisi vaatinut huomattavasti enemmän verkkosynkronointia ja lisätyötä verrattuna projektin tavoitteisiin.

Verkkotoiminnallisuuksien testaamista varten projektiin toteutettiin yksinkertainen käyttöliittymä, jonka avulla pelaaja pystyi luomaan palvelimen tai liittymään olemassa olevaan peliin painikkeiden kautta. Toteutus mahdollisti kahden pelaajan Neljän suora -pelin käynnistämisen palvelimen kautta suhteellisen helposti. Kehityksen aikana Unityn Console-ikkunaa hyödynnettiin aktiivisesti ohjelmakoodin toiminnan ja eri verkkotoimintojen tarkistamiseen.

Unity Netcode for GameObjects -version valmistumisen jälkeen projekti siirrettiin Photon-verkkoratkaisuun. Tässä vaiheessa haasteena oli poistaa Unity Netcode for GameObjects -toteutukseen liittyvät verkkotoiminnallisuudet ja korvata ne Photonin omilla RPC-ratkaisuilla. RPC-kutsujen toimintalogiikan muuttaminen vaati uudelleen perehtymistä Photon PUN 2 -verkkoratkaisun toimintaperiaatteisiin.

Photon-version toteutusta helpotti aikaisempi kokemus Photon PUN 2 -verkkoratkaisun käytöstä toisessa projektissa. Aiempi kokemus helpotti erityisesti PhotonView-komponenttien ja RPC-kutsujen käyttöönottoa.

Photon-version käyttöönotossa verkkopalvelin luotiin Photonin selainpohjaisen hallintapaneelin kautta, minkä jälkeen palvelimen tiedot yhdistettiin Unity-projektiin. Tämä ratkaisu erosi Unity Netcode for GameObjects -toteutuksesta, jossa verkkoyhteyden hallinta toteutettiin suoraan pelimoottorin sisällä. Muutoin

verkkototeutusten toimintaperiaatteet olivat suhteellisen samankaltaisia, vaikka RPC-kutsujen toteutustapa erosikin verkkoratkaisujen välillä.

Photon-version jälkeen projektia kokeiltiin siirtää Alteruna-verkkoratkaisuun. Alterunan käyttöönotossa hyödynnettiin aluksi valmiita huonejärjestelmä- ja prefab-ratkaisuja, mutta niiden käyttöönotossa ilmeni ongelmia. Tämän vuoksi toteutuksessa siirryttiin yksinkertaisempaan host- ja join-ratkaisuun, jossa pelaajat pystyivät muodostamaan yhteyden toisiinsa lähiverkon kautta.

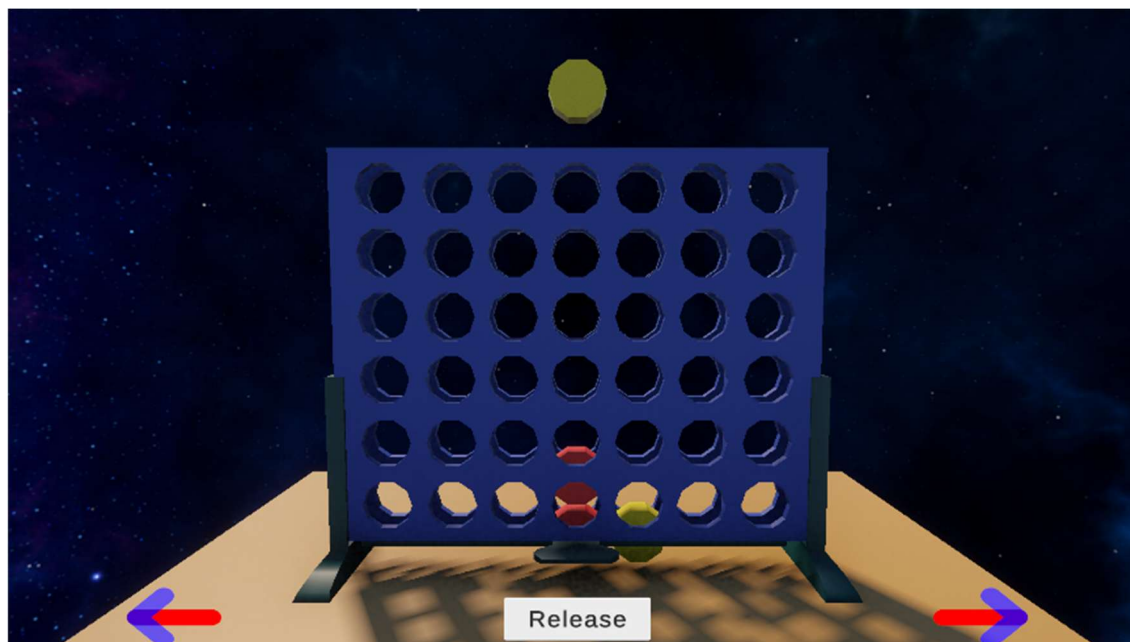
Alteruna-version toteutuksen aikana havaittiin, että yhteyden muodostaminen ja pelilogiikan synkronointi eivät olleet yhtä suoraviivaisia kuin Unity Netcode for GameObjects- ja Photon-versiossa. Vaikka pelaajat saatiin yhdistettyä samaan verkkosessioon, Neljän suora -pelin siirtojen synkronointi ei toiminut luotettavasti molempien pelaajien välillä. Tämän vuoksi Alteruna-version kehitys jäi osittain keskeneräiseksi, mutta se tarjosi silti hyödyllisen vertailukohdan verkkoratkaisujen käyttöönoton ja kehitysprosessin arviointiin.

4.3 Kehitysvaiheen keskeiset havainnot

Projektin kehityksen aikana havaittiin useita ja positiivisia että haastavia asioita. Suurin osa kehitystyön aikana ilmenneistä ongelmista liittyi ohjelmakoodin virheisiin ja verkkototeutusten toiminnan tarkistamiseen. Virheiden etsiminen ja korjaaminen oli merkittävä osa projektin kehitysprosessia.

Projektin aikana ohjelmointiosaaminen, verkkoratkaisujen ja -toteutusten osalta kehittyi huomattavasti. Unity Netcode for GameObjects -verkkoratkaisun käyttö toi uuden näkökulman moninpelien toteuttamiseen, sillä suuri osa verkkotoiminnallisuuksista voitiin toteuttaa suoraan Unity-pelimoottorin sisällä ilman erillisiä selainpohjaisia palvelinratkaisuja.

Yksi projektin yllättävimmistä ongelmista liittyi nappuloiden sijoittumiseen peli-
laudalla. Oletuksena oli, että nappulat sijoittuisivat automaattisesti colliderin
keskelle, mutta käytännössä nappulat sijoittuivat colliderin alareunaan. Kuva 4
havainnollistaa ongelmatilannetta.



Kuva 4. Nappulan sijainti väärin laudalla.

Ongelmaa pyrittiin ratkaisemaan muokkaamalla colliderin ja prefab-objektien
asetuksia, mutta lopullinen ratkaisu toteutettiin muuttamalla nappuloiden sijain-
tia ohjelmallisesti. Vaikka ratkaisu ei ollut teknisesti kaikkein optimaalisin, se
mahdollisti pelin toiminnallisuuden toteuttamisen projektin tavoitteideni mukai-
sesti.

Verkkototeutusten vaihtaminen Unity Netcode for GameObjects -ratkaisusta
Photoniin ja siitä Alterunaan toi myös omia haasteita. Verkkoratkaisujen RPC-
kutsut toimivat eri tavoin, minkä vuoksi verkkotoiminnallisuuksien verkkoratkai-
sujen toimintaperiaatteet olivat osittain samankaltaisia, mutta niiden tekninen to-
teutus erosi huomattavasti toisistaan.

Alterunan toteutuksessa havaittiin useita haasteita erityisesti verkkosynkronoinnissa. Alterunan mukana tulevia valmiita prefab- ja huonejärjestelmäratkaisuja kokeiltiin projektin aikana, mutta niiden käyttöönotossa ilmeni ongelmia eikä niitä saatu toimimaan luotettavasti projektin tarpeisiin. Tämän vuoksi projektissa päädyttiin toteuttamaan omat käyttöliittymäpainikkeet ja yksinkertaisempi host- ja join-ratkaisu.

Vaikka verkkoyhteys pelaajien välillä saatiin muodostettua onnistuneesti, pelin siirtojen täydellinen synkronointi osoittautui haastavammaksi kuin Unity Netcode for GameObjects- ja Photon-toteutuksissa. Erityisesti synkronoitavien objektien hallinta ja Alterunan omat verkkokomponentit vaativat enemmän perehtymistä kuin projektin aikataulu mahdollisti. Tämän vuoksi Alteruna-version verkkosynkronointi jäi osittain keskeneräiseksi verrattuna muihin toteutuksiin.

Projektin aikana havaittiin, että Alteruna soveltui paremmin valmiisiin verkkoratkaisuihin pohjautuviin toteutuksiin, kun taas Unity Netcode for GameObjects ja Photon mahdollistivat suoraviivaisemman ohjelmallisen hallinnan Neljän suora - projektin kaltaisessa yksinkertaisessa moninpelissä.

Projektin aikana tuli myös vastaan tilanteita, joissa ongelmien ratkaiseminen vaati itsenäistä tiedonhakua virallisen dokumentaation ulkopuolelta. Tällaisissa tilanteissa ratkaisua etsittiin dokumentaatiosta, ohjelmointifoorumeilta ja tekoälypohjaisista työkaluista, kuten ChatGPT:stä ja GitHub Copilotista. Erityisesti verkkototeutuksiin liittyvissä ongelmatilanteissa tekoäly helpotti yksittäisten ohjelmointivirheiden tunnistamista ja vaihtoehtoisten ratkaisujen löytämistä silloin, kun vastaavaa ongelmaa ei löytynyt suoraan muista verkkolähteistä.

5 Yhteenveto

Tämän opinnäytetyön tavoitteena oli vertailla eri verkkoratkaisuja Unity-pelimoottorissa käytännön peliprojektin avulla. Työssä toteutettiin Neljän suora - monipeli käyttäen Unity Netcode for GameObjects-, Photon- ja Alteruna-verkkoratkaisuja. Tarkoituksena oli tutkia verkkoratkaisujen käyttöönottoa, käytettävyyttä, kehitysprosessia ja verkkosynkronoinnin toteutusta käytännössä.

Projektin aikana havaittiin, että verkkoratkaisujen välillä oli selkeitä eroja niin käyttöönotossa kuin ohjelmointitavoissakin. Unity Netcode for GameObjects integroitui hyvin Unity-pelimoottoriin ja tarjosi selkeän tavan toteuttaa verkkosynkronointia Unityn omien verkkokomponenttien avulla. Photon puolestaan tarjosi toimivan ja suoraviivaisen RPC-pohjaisen verkkoratkaisun, jonka käyttöönottoa helpotti valmis verkkopalvelu ja kattava dokumentaatio.

Alteruna erosi kahdesta muusta verkkoratkaisusta selvästi. Sen toteutus perustui enemmän valmiisiin komponentteihin ja prefab-ratkaisuihin, minkä vuoksi verkkorakenteen ymmärtäminen osoittautui haastavammaksi. Projektissa verkko-yhteyden muodostaminen onnistui, mutta pelin siirtojen täydellistä synkronointia ei saatu toteutettua samalla tasolla kuin Unity Netcode for GameObjects- ja Photon-versioissa.

Työn aikana havaittiin myös, kuinka tärkeää verkkoratkaisun valinta on peliprojektin kehityksen kanalta. Eri verkkoratkaisut soveltuvat erilaisiin projekteihin riippuen pelin laajuudesta, verkkorakenteesta ja kehittäjän osaamisesta. Lisäksi työ osoitti, että dokumentaation laatu ja valmiiden esimerkkien saatavuus vaikuttavat merkittävästi verkkoratkaisun oppimisen ja käyttöönottoon.

Jatkokehityksenä projektia voisi laajentaa esimerkiksi lisäämällä enemmän huoneita, jolla kasvatetaan pelaajia, kehittämällä käyttöliittymää tai toteuttamalla verkkototeutuksille tarkempia suorituskykytestejä. Lisäksi Alteruna-version verkkosynkronointia voisi jatkokehittää toimivammaksi perehtymällä tarkemmin sen synkronointijärjestelmään ja verkkokomponentteihin.

Lähteet

- 1 Vigier, Pierre. 2019. Beginner's Guide to Game Networking. Verkkoaineisto. pvigier.github.io. <<https://pvigier.github.io/2019/09/08/beginner-guide-game-networking.html>>. 8.9.2019. Luettu 21.3.2026.
- 2 Netcode for GameObjects. Verkkoaineisto. Unity Technologies. <<https://docs.unity3d.com/Packages/com.unity.netcode.gameobjects@2.11/manual/index.html>>. Luettu 21.3.2026.
- 3 Get started with Netcode for GameObjects. Verkkoaineisto. Unity technologies. <<https://learn.unity.com/tutorial/get-started-with-netcode-for-gameobjects>>. Luettu 21.3.2026.
- 4 RPC. Verkkoaineisto. Unity Technologies. <<https://docs.unity3d.com/Packages/com.unity.netcode.gameobjects%402.5/manual/advanced-topics/message-system/rpc.html>>. Luettu 23.3.2026.
- 5 Photon PUN Introduction. Verkkoaineisto. Photon. <<https://doc.photonengine.com/pun/current/getting-started/pun-intro>>. Luettu 8.5.2026.
- 6 Matchmaking Guide. Verkkoaineisto. Photon. <<https://doc.photonengine.com/pun/current/lobby-and-matchmaking/matchmaking-and-lobby>>. Luettu 8.5.2026.
- 7 RPCs and RaiseEvent. Verkkoaineisto. Photon. <<https://doc.photonengine.com/pun/current/gameplay/rpcsandraiseevent>>. Luettu 8.5.2026.
- 8 Pricing and Licensing. Verkkoaineisto. Photon. <<https://www.photonengine.com/pun/pricing>>. Luettu 8.5.2026.
- 9 Beginner User Guide. Verkkoaineisto. Alteruna. <https://docs.v2.alteruna.com/html/T_Alteruna_Guides_BeginnerUserGuide.htm>. Luettu 14.5.2026.
- 10 Synchronizable Class. Verkkoaineisto. Alteruna. <https://alteruna.github.io/au-multiplayer-api-docs/html/T_Alteruna_Synchronizable.htm>. Luettu 14.5.2026.
- 11 SimonAlteruna. Multiplayer-sdk-unity-package. Verkkoaineisto. <<https://github.com/Alteruna/multiplayer-sdk-unity-package>>. Luettu 14.5.2026.
- 12 Galactic Kittens. Verkkoaineisto. Unity Technologies. <<https://unity.com/demos/galactic-kittens>>. Luettu 4.6.2026.

- 13 Boss Room: Small scale co-op sample. Verkkoaineisto. Unity technologies. <<https://unity.com/demos/small-scale-coop-sample>>. Luettu 4.6.2026.
- 14 Maldonado, Esteban. 2022. Building my first multiplayer game with Netcode for GameObjects. Verkkoaineisto. <<https://unity.com/blog/games/building-a-multiplayer-game-first-time-netcode-for-gameobjects>>. 26.10.2022. Luettu 4.6.2026.
- 15 Phasmophobia. Verkkoaineisto. Kinetic Games. <<https://kineticgames.co.uk/phasmophobia>>. Luettu 4.6.2026.
- 16 Samples. Verkkoaineisto. Photon. <<https://www.photonengine.com/samples>>. Luettu 4.6.2026.
- 17 Showcase. Verkkoaineisto. Alteruna <<https://www.alteruna.com/showcase>>. Luettu 4.6.2026.
- 18 SimonAlteruna. Multiplayer FPS Template. Verkkoaineisto. <<https://github.com/Alteruna/Multiplayer-FPS-Template>>. Luettu 4.6.2026.
- 19 SimonAlteruna. Multiplayer Racing Template. Verkkoaineisto. <<https://github.com/Alteruna/Multiplayer-Racing-Template>>. Luettu 4.6.2026.
- 20 Solvern. 2021. Connect 4. Verkkoaineisto. <<https://sketchfab.com/3d-models/connect-4-90bc2c11ed694147be30413c3ac05de7>>. 21.4.2021. Luettu 21.3.2026.