Joonas Sallinen

Moninpeliratkaisun toteutus olemassa olevaan Unity-peliin

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

11.11.2024

Tiivistelmä

Tekijä:	Joonas Sallinen
Otsikko:	Moninpeliratkaisun toteutus olemassa olevaan Unity-peliin
Sivumäärä:	41 sivua
Aika:	11.11.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Pelisovellukset
Ohjaajat:	Lehtori Heini Puuska

Insinöörityön tavoitteena oli toteuttaa moninpelimuoto Unity-pelimoottorilla luotuun peliprojektiin. Moninpelimuodon piti lisäksi olla yhteensopiva Steam-alustan kanssa.

Insinöörityössä tutustuttiin ensin Fast and Fluffy -peliprojektin keskeiseen sisältöön. Seuraavaksi perehdyttiin verkkomalleihin, joiden avulla pystyy kuvaamaan, miten yhteys luodaan pelaajien ja palvelimien välillä. Moninpelin toteuttamista varten valittiin asiakas-isäntä-verkkomalli (client-host), jossa yksi pelaaja toimii pelin isäntänä, eli hänen laitteensa simuloi peliä, välittää pelitilanteen muille pelaajille ja tekee päätöksen pelitilanteesta.

Verkkomallin valinnan jälkeen etsittiin verkkoratkaisu, jonka avulla saataisiin toteutettua valitun verkkomallin mukainen toiminta. Projektityön tekemistä varten valittiin Mirror-verkkokirjasto, jonka tarkoitus on helpottaa verkkotoiminallisuuden luontia Unity-pelimoottorille.

Lopputuloksena saatiin aikaan moninpelimuoto, jossa pelaajat voivat joko luoda aulan tai liittyä olemassa olevaan aulaan. Pelaaja, joka luo aulan, toimii isäntänä, ja muut pelaajat yhdistyvät isännän laitteeseen pelin sisäisen palvelinlistan kautta tai Steam-alustan ystävälistan kautta.

Pelissä pelaajien ja esineiden toiminnat on synkronoitu verkon kautta hyödyntämällä Mirror-verkkoratkaisun verkkoattribuutteja, koodia ja komponentteja. Pelitilanteen tiedot välitetään pelaajille Steamworks-välityspalvelimen avulla, joka pystyy välittämään pelitietoja isännän ja muiden pelaajien välillä. Välityspalvelin kuuluu Steamworks-rajapinnan sisältöön. Sen hyödyntämisen mahdollistaa Mirror-verkkokirjaston FizzySteamworks-kuljetuskerros.

Projektityön toteutus täytti asetetut tavoitteet. Toteutuksessa yksi isoimmista haasteista oli verkkomoninpelaamisen testaaminen kahdella pelaajalla, koska pelitiedot piti siirtää toiselle laitteelle ja Steam-käyttäjälle, mikä vei paljon aikaa. Insinöörityössä tuli tutuksi verkkomallit, verkkoratkaisut ja se, miten verkkomoninpelit toimivat varsinkin Unity-pelimoottorissa.

Avainsanat: pelikehitys, moninpeli, verkkomoninpeli, Steam, Unity

---

Abstract

Author:	Joonas Sallinen
Title:	Implementation of a multiplayer solution to an existing Unity game
Number of Pages:	41 pages
Date:	11 November 2024
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Game Applications
Supervisors:	Heini Puuska, Senior Lecturer

The goal of the project was to implement an online multiplayer mode for a game project created with the Unity game engine and to make it compatible with the Steam platform.

The project began by reviewing the core content of the Fast and Fluffy game project. Different network models were then studied to understand how connections between clients and servers can be created. The Client-Host model, where one player acts as the server, was chosen for the implementation. After choosing the network model, network solutions were studied in order to find a solution that could implement the desired end result, and the Mirror networking library was chosen.

The result was an online multiplayer mode where players can create or join lobbies. The lobby creator acts as the host and other players connect to their device using the in-game server list or the Steam friends list. Player actions and game objects were synchronized with Mirror's network attributes and components. Game state data was relayed using the relay server of the Steamworks API, which was accessed using Mirror's FizzySteamworks transport.

The project successfully met its goals. The thesis provides an understanding of network models, solutions, and the implementation of an online multiplayer, particularly in Unity.

Keywords: game development, multiplayer, online multiplayer, Steam, Unity

Sisällys

1 Johdanto

Insinöörityössä tutkitaan, miten moninpelimuoto voitaisiin toteuttaa olemassa olevaan Unity-pelimoottorissa luotuun Fast and Fluffy -peliprojektiin. Fast and Fluffy on kilpailupeli, jossa pelaajat kilpailevat siitä, kuka kiertää pelitason radan kolme kertaa nopeimmin. Pelissä pelaajat pystyvät liikkumisen lisäksi sivuluisua ja käyttää radalle ilmestyviä esineitä. Pelissä on jo toteutettu paikallispelimuoto, jossa pelaajat voivat pelata yhdellä laitteella jaetulla näytöllä.

Projektityön tavoitteena on toteuttaa moninpelimuoto, jossa pelaajat pystyvät pelaamaan verkon kautta yhdessä. Moninpelimuodon pitää toimia Steamworks-rajapinnan (API) kanssa, sillä peli mahdollisesti julkaistaan Steam-alustalle. Insinöörityössä esitellään moninpelimuodon toteuttamiseen liittyviä teknologioita, jotta pelikehittäjät voivat hyödyntää raporttia oman moninpelin toteuttamiseen varsinkin, jos moninpelin halutaan toimivan Steam-alustan kanssa.

Steam-alusta on Valve-yrityksen hallinnoima verkossa oleva videopelien jakelupalvelu ja myymälä. Steam-alustalla voi ostaa pelejä ja verkostoitua muiden pelaajien kanssa. Steamworks on rajapinta, jonka avulla pelinkehittäjät voivat käyttää Steam-alustaan liittyviä toimintoja. Steamworks-palveluun kuluu esimerkiksi välittäjäpalvelin ja funktioita, joiden avulla voidaan hakea Steam-alustassa olevia tietoja, kuten Steam-käyttäjätietoja ja pelipalvelimien tietoja.

Insinöörityössä tutustutaan ensin Fast and Fluffy -peliprojektin keskeiseen sisältöön. Tämän jälkeen tutkitaan verkkomalleja ja verkkoratkaisuja ja valitaan sopivat moninpelimuodon toteuttamiseen. Lopuksi käydään läpi moninpelimuodon toteuttaminen valittujen verkkoteknologien avulla. Projektityön lopputulos tulee mahdollistamaan useamman pelaajan yhdessä pelaamisen verkon kautta Steam-alustan avulla.

2 Fast and Fluffy -peliprojekti

Projektityön moninpelaaminen toteutetaan olemassa olevaan peliprojektiin Fast and Fluffy, joka on Unity-pelimoottorilla luotu kilpailupeli tietokoneelle. Pelissä pelaajat kilpailevat siitä, kuka kiertää kilpailuradalla kolme kertaa nopeimmin. Pelaajat hallitsevat pelissä heitä edustavia pelaajahahmoja, jotka pystyvät liikkumaan juoksemalla ja sivuluisumalla. Kilpailu tapahtuu pelitasossa, joka koostuu sarjakuvamaiseksi ja söpöksi tyylitellystä ympäristöstä ja kilpailuradasta, jota pelaajat seuraavat.

Pelissä on valmiiksi toteutettu paikallinen moninpelaaminen, jossa neljä pelaajaa voi pelata samanaikaisesti jaetulla näytöllä yhdellä laitteella. Pelaajat voivat pelin aikana vaikuttaa pelitilanteeseen esineiden (power ups) avulla, joita voi käyttää hyökkäämiseen, puolustamiseen tai oman pelaajahahmon vauhdin lisäämiseen.

2.1 Päävalikko ja pelitasot

Pelissä on päävalikko ja pelitasoja, jotka ovat Unity-pelimoottorissa scene-tiedostoja. Scene-tiedostot sisältävät pelitasoon liittyviä tietoja, kuten objektien sijainteja, skaaloja ja komponentteja, jotta niitä voidaan esitellä 3D-tilassa. Unity-dokumentaation mukaan jokaista scene-tiedostoa voi ajatella omana pelitasonaan. (1.) Päävalikon ja pelitasojen välillä siirrytään, kun kilpailu aloitetaan ja lopetetaan. Raportin kirjoituksen aikana pelissä oli vain yksi pelitaso.

Päävalikko

Päävalikko (kuva 1) avautuu, kun peli käynnistetään. Päävalikosta pystyy aloittamaan pelin, muuttamaan asetuksia ja sulkemaan pelin. Päävalikko toimii lisäksi peliaulana, johon pelaajat ilmestyvät, kun pelilaitteeseen yhdistetyllä ohjaimella painetaan liittymispainiketta. Aulaan luodut pelaajahahmot asetellaan näkyviin valikon vieressä olevaan 3D-tilaan. Pelin voi aloittaa painamalla Arcade-nappia, jolloin pelitason lataaminen aloitetaan.

Kuva 1. Pelin päävalikko (2).

Pelitasot

Pelitasot sisältävät ympäristön ja kilpailuradat (kuva 2), jossa kilpailu pelataan. Kilpailuradalla on esinepalloja ja tarkastuspisteitä. Esinepallot antavat pelaajille esineitä ja tarkastuspisteiden avulla varmistetaan, että koko kilpailurata on käyty läpi ennen maalin läpi menemistä.

Kuva 2. Pelitason kilparata (2).

Pelitasoon siirryttyä suoritetaan alkuanimaatio, jossa peliratta esitellään. Alkuanimaation aikana pelaajat asetetaan aloitusviivan taakse (kuva 3). Pelaajien liikkuminen on pysäytetty, kunnes kilpailu alkaa. Peliradan esittelyn jälkeen aloitetaan aloitusajastin, jonka jälkeen kilpailu alkaa ja pelaajat pystyvät liikkumaan.

Kuva 3. Pelintason aloituspaikka (2).

Pelitasoissa on eri reittejä ja salaisia oikopolkuja, joiden avulla pelaajat voivat saada etumatkaa muihin. Esimerkiksi kuvissa 2 ja 3 esitelty pelitaso sisältää salaisen oikopolun ennen aloitusviivalle takaisin saapumista; jos menee pensaiden läpi voi välttää pidemmän polun.

2.2 Pelaajahahmot ja esineet

Pelaajat hallitsevat omaa pelaajahahmoaan (kuva 4) joko konsoliohjaimella tai näppäimistöllä. Nämä pelaajahahmot pystyvät liikkumaan, sivuluisumaan ja käyttämään esineitä pelissä. Pelaajahahmot luodaan jokaiselle liittyneelle pelaajalle aulassa.

Kuva 4. Pelaajahahmo-objekti (2).

Sivuluisuminen on tapa lisätä nopeutta pelaajahahmolle käännön aikana. Pelaaja pystyy sivuluisumaan liikkumalla sivusuuntaan ja samaan aikaan painamalla sivuluisu-nappia, joka on näppäimistöllä välilyöntinäppäin ja ohjaimella alapainike.

Pelaajahahmolla voi olla vain yksi esine kerrallaan, ja kannossa oleva esine pitää käyttää ennen uuden hankkimista. Pelaajien näytöillä (kuva 5) esitetään pelaajaan liittyvää tietoa. Näytön oikeassa alakulmassa näkyy pelaajan sijainti eli missä pelaaja on verrattuna muihin pelaajiin radalla. Oikeassa yläkulmassa näkyy kierrosluku eli kuinka monta kierrosta pelaaja on suorittanut. Vasemmassa yläkulmassa näkyy hallussa olevan esineen ikoni.

Kuva 5. Pelinäkymä pelaajan näkökulmasta (2).

Esineiden (power ups) avulla pelaaja voi vaikuttaa omaan tai muiden pelaajien pelaajahahmoihin. Pelaajat saavat haltuunsa esineitä liikkumalla esinepallon (kuva 6) läpi, jolloin pelaaja saa satunnaisesti yhden esineen. Esinepallot ilmestyvät radalle neljän ryhmissä riveittäin tietyissä paikoissa.

Kuva 6. Pelitasossa neljä esinepalloa kilpailuradalla rivissä (2).

Esineet pystyvät joko tainnuttamaan, suojaamaan tai lisäämään pelaajahahmon nopeutta. Hallussa olevaa esinettä käytetään painamalla esineenkäyttönappia, joka on näppäimistöllä E-näppäin tai ohjaimella vasen painike. Jokaista esinettä voi käyttää lisäksi eteenpäin tai taaksepäin, jolloin esineen efekti suuntautuu eri tavalla. Tämä toimii näppäimistöllä ylä- ja ala-nuolinäppäimillä tai ohjaimella ylä-painikkeella ja oikealla painikkeella. Esimerkiksi Toy Mallet -esine normaalisti käyttäen lyö vasaran maahan ja tainnuttaa käyttäjän lähellä olevat pelaajahahmot hetkeksi, mutta jos vasaraa käytetään yläsuuntaan se lähettää eteenpäin paineaallon, joka tainnuttaa pelaajia.

2.3 Paikallinen moninpelaaminen

Pelissä olevassa paikallisessa moninpelissä useampi, kuin yksi pelaaja voi pelata yhdellä laitteella. Pelaajat pystyvät liittymään peliin päävalikossa painamalla liittymisnappia. Pelaajan pelaajahahmo ilmestyy aulaan (kuva 7) näkyviin, kun hän on liittynyt.

Kuva 7. Päävalikon aula, johon on liittynyt kaksi pelaajaa (2).

Pelitasossa näyttö jaetaan (kuva 8) pelaajamäärän mukaan. Pelaajia voi olla paikallisessa moninpelimuodossa enintään neljä, jotta näytön jakaminen

näyttäisi järkevältä. Pelaajahahmoille asetetaan eri värit, jotta ne voidaan erottaa toisistaan pelin aikana.

Kuva 8. Pelinäkymä, jossa näyttö on jaettu kahden pelaajan välillä (2).

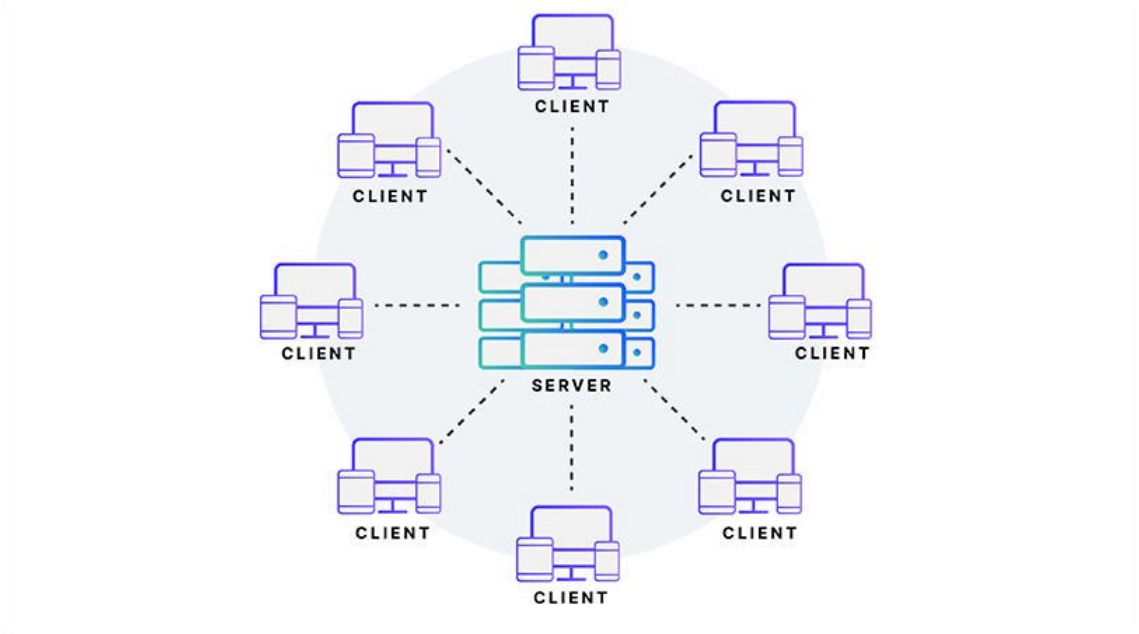
Paikallisessa pelimuodossa peli loppuu, kun jokainen pelaaja on kiertänyt radan kolme kertaa ja saapunut maaliin. Pelin lopussa pelaajien ajat esitetään heidän saapumisjärjestyksessään.

3 Verkkomallit

Moninpelin toteuttamista varten voidaan hyödyntää moninpeliverkkomalleja, joiden avulla voidaan hahmottaa arkkitehtuuria moninpelin toimintaa varten. Yleisimmät verkkomallit ovat asiakas-palvelin (client-server) ja vertaisverkko (peer-to-peer) (3; 4; 5).

3.1 Asiakas-palvelin

Asiakas-palvelin-verkkomallissa (kuva 9) asiakkaat (client) yhdistäytyvät palvelimeen (server), joka simuloi peliä ja käsittelee asiakkaiden syötteet. Palvelimeen yhdistyneet asiakkaat simuloivat omaa versiota pelistä laitteillaan, joihin palvelimella simuloidusta pelistä lähetetään tietoa, jotta asiakaspelaajan laite voi päivittää omaa pelisimulaatiotaan. (3.)

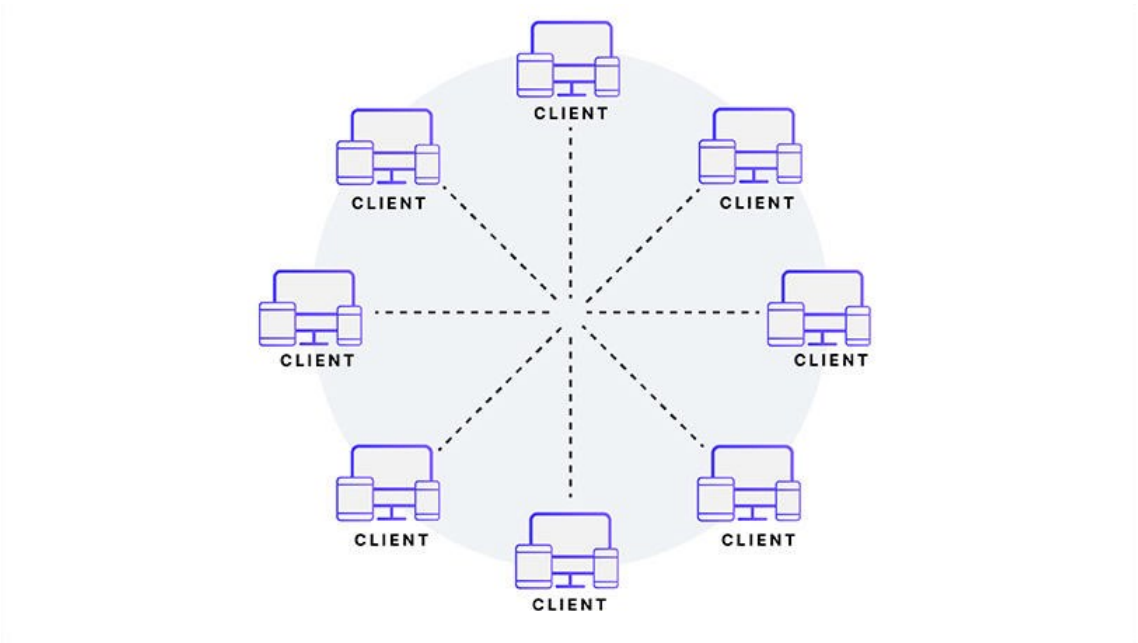


Kuva 9. Asiakas-palvelin verkkomallin yhteydet esitettynä kuvassa (4).

Asiakas-palvelin-verkkomallissa pitää ottaa huomioon palvelimen ylläpidon hinta ja mahdollinen tarve palvelun laajentamiseen, jos pelaajamäärä kasvaa (6).

3.2 Vertaisverkko

Vertaisverkkomallissa (kuva 10) asiakkaat ovat yhdistyneet suoraan toisiinsa, joten jokaisen pelaajan laite hoitaa pelilogiikan suorittamista ja pelitiedon jakamista muille laitteille pelin tilan päivittämistä varten. Toisin sanoen jokainen pelaaja toimii kuin palvelin asiakas-palvelin-verkkomallissa. (3.)

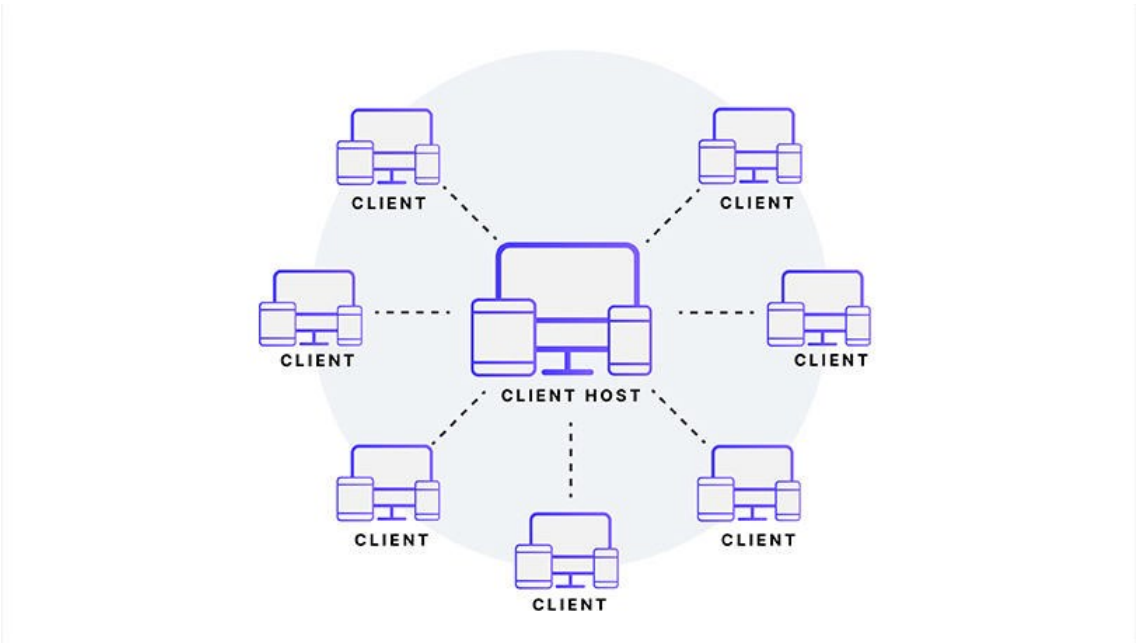


Kuva 10. Vertaisverkkomallin yhteydet esitettynä kuvassa (3).

Vertaisverkkomallissa tulee esiin ongelmia yhdistämisen ja turvallisuuden kanssa. Ilman välityspalvelinta reititin voi estää tiedon välittämisen, jolloin pelaajat joutuvat avaamaan portin reitittimessä. Kun portti on auki, henkilöt, jotka tietävät tai arvaavat avatun portin numeron, voivat lähettää mahdollisesti haitallisia tietopaketteja laitteelle. (6.)

3.3 Asiakas-isäntä

Asiakas-isäntä-verkkomalli (Client-Host) (kuva 11) tai kuunteleva palvelin (Listen Server) on malli, jossa yksi asiakaslaite toimii isäntänä eli palvelimena ja pelaajana. Isäntä-pelaajan laite simuloi peliä, tekee päätöksen pelitilanteesta ja välittää pelitilanteen muille pelaajille. (3.)



Kuva 11. Asiakas-isäntä verkkomallin yhteydet esitettynä kuvassa (3).

Asiakas-isäntä-verkkomallissa on sama ongelma kuin vertaisverkkomallissa, jos ei käytetä välityspalvelinta. Lisäksi pelaajalla, jonka laite toimii isäntänä, tulee olemaan etu peleissä, sillä isännän laite hallitsee pelin tilannetta (6).

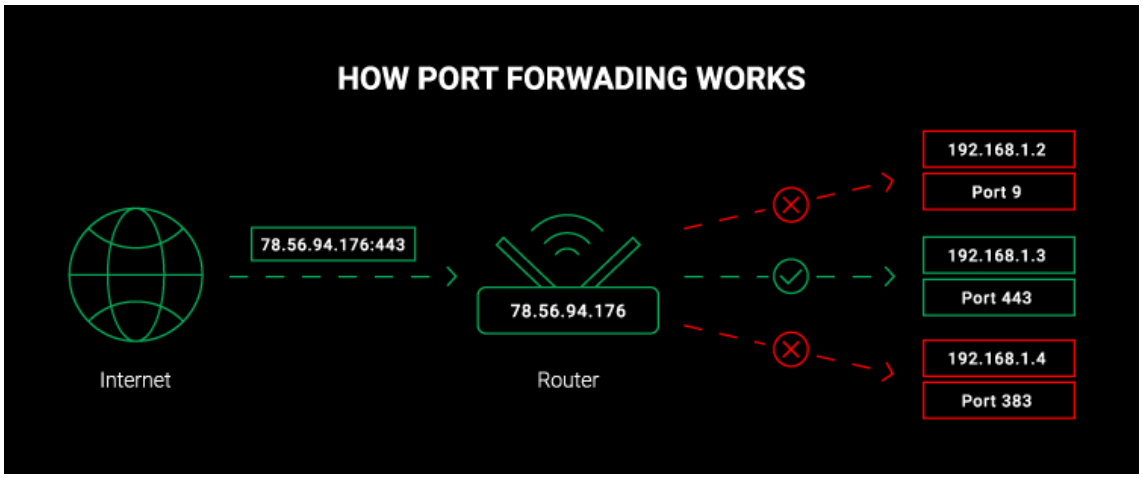
3.4 Pelitiedon välittäminen

Pelin tiedon välittämisen yleisimmät tavat ovat palvelimen kautta, välityspalvelimen (relay server) kautta tai suoralla yhteydellä asiakaslaitteiden välillä. Palvelin simuloi peliä ja välittää tämän simuloidun pelitilanteen asiakaslaitteille. Ilman palvelinta pelin tiedon siirtäminen pitää tehdä joko välityspalvelimen avulla tai suoraan yhdistämällä asiakkaiden laitteet. (3; 7.)

Välityspalvelin välittää pelkästään tietoa, joten pelissä pitää määritellä, miten pelitilanne päätetään. Asiakas-isännöidyllä mallilla pelaaja, jonka laite isännöi pelin, on lopullinen päättäjä pelitilanteesta. Vertaisverkkomallissa ei ole pelitilanteen päättäjää, vaan kaikki asiakkaat toimivat pelaajina sekä välittävät tietoa, eli jokaisen asiakkaan pelitilanne välitetään kaikille muille pelaajille ja kaikki pelaajien laitteet päivittävät oman pelitilanteensa sen mukaan. (3; 7; 8.) Välityspalvelin helpottaa pelaajien yhdistämistä, sillä pelaajat voivat yhdistyä välityspalvelimeen ilman, että palomuuri tai osoitteenmuunnos-teknologia (Network Address Translation) estää yhteyden (6).

Osoitteenmuunnos on reitittimissä yleisesti oleva teknologia, jonka avulla reititin suojaa sisäverkossa olevien laitteiden IP-osoitteita. Se tekee tämän vaihtamalla sisäverkosta pyyntöä lähettävän laitteen IP-osoitteen reitittimen omaan julkiseen IP-osoitteeseen. Tällöin ulkoiset laitteet eivät voi luoda suoraa yhteyttä sisäverkossa olevien laitteiden kanssa. (6.) Jotta pelaajia voi yhdistää muiden pelaajien laitteisiin verkon kautta ilman palvelinta, joko asiakas-isäntä tai vertaisverkkomallin mukaan, pelaajien pitää tehdä portinohjaus (Port Forwarding) (9; 10).

Portinohjaus (kuva 12) on tapa antaa ulkoisten laitteiden muodostaa suora yhteys sisäverkossa oleviin laitteisiin, vaikka laitteet olisivat palomuurin tai reitittimen suojauksessa. Portinohjaus toimii kertomalla reitittimelle avoimen portin numerosta, jota se käyttää tuntemattomien yhteyksien välittämiseen. Portin numeron avulla reititin voi välittää tuntemattomat yhteydet, joilla on portin numero, laitteelle, joka avasi portin. (9; 10; 11.)



Kuva 12. Esimerkki portinohjaamisen toiminnasta (12).

Portin avaamisessa on kuitenkin turvallisuusriskejä, sillä ulkoiset laitteet, jotka tietävät portin numeron, voivat lähettää paketteja tietoa tai jopa ottaa täyden hallinnan laitteesta avatun portin kautta ilman turvallisuustarkistusta (11).

3.5 Verkkomallin valinta

Projektityön toteuttamista varten valittiin asiakas-isäntä-verkkomalli, koska siinä pelin isäntä voi toimia lopullisena päätöksentekijänä pelitilanteissa ja verkkomallille löytyi aiheeseen sopivia verkkoaineistoja. Asiakas-isäntä-verkkomalli on myös halvempi, sillä ei tarvitse ylläpitää tai ostaa omaa palvelinta. Koska pelin halutaan toimivan Steam-alustassa, voidaan hyödyntää Steamworks-rajapinnan välityspalvelinta, jolloin ei tarvitse hankkia omaa palvelinta, vaan pelaajat voivat yhdistyä peliin Steam-alustan avulla.

4 Verkkoratkaisut

4.1 Verkkoratkaisujen sisältö

Verkkoratkaisut ovat työkalujapaketteja, jotka sisältävät verkon kautta pelaamiseen liittyviä toimintoja, joiden tarkoituksena on helpottaa moninpelin luontia. Verkkoratkaisut yleensä sisältävät kuljetuskerroksen (Transport Layer), joka huolehtii tiedonvälityksestä ja korkean tason abstraktiokerroksen (Netcode), joka sisältää moninpelin luomista varten yleisiä toimintoja ja ominaisuuksia. (13.)

Korkean tason abstraktiokerroksella tai verkkokoodilla tarkoitetaan moninpelin toiminnan toteuttamistaan varten olevia ominaisuuksia ja valmiiksi luotuja funktioita esimerkiksi pelaajien liittymiseen, pelin isännöimiseen ja pelitilanteen hallitsemiseen (13). Esimerkiksi Mirror-verkkoratkaisu sisältää erilaisia kuljetuskerroksia, tiedonvälitystä, tilanteen synkronointia, valtuutuksen hallintaa ja asiakas-palvelin-arkkitehtuurin tukea.

Tiedonvälityksellä tarkoitetaan tapoja lähettää ja vastaanottaa tietoa sekä reagoida välitettyyn tietoon, esimerkiksi pelitilanteen muutoksiin, pelaajan liittymiseen ja viestien lähetykseen (14). Synkronoinnilla tarkoitetaan skriptien muuttujien arvojen (kokonaisluvut, liukuluvut, merkkijonot ja booleanit) muutoksien jakamista palvelimelta asiakaslaitteille (15).

Valtuutus on tapa kertoa, kuka omistaa objektin pelissä eli millä laitteella on valtuus vaikuttaa tähän objektiin. Laite, jolla on valtuus, voi olla pelaaja tai palvelin. Oletusarvoisesti palvelimella on valtuus kaikkiin objekteihin ja pelaajille asetetaan tarvittaessa valtuus heidän pelaajahahmoihinsa, jolloin he pystyisivät hallitsemaan niiden toimintaa. (16.)

Arkkitehtuurin tukemisella tarkoitetaan, että verkkoratkaisu tukee verkkomallin toteuttamista. Esimerkiksi Mirror-verkkokirjasto on palvelin-valtuutettu eli toiminnot simuloidaan palvelimella, mutta Mirror tukee sitä, että asiakaslaite voi toimia

pelaajana ja palvelimena samaan aikaa, joten se tukee asiakas-palvelin-verkkomallia (17).

Kuljetuskerros hallitsee tiedonvälityksen asiakkaiden ja isännän tai palvelimen välillä. Yleensä tiedon siirto tehdään joko TCP- tai UDP-tietoliikenneprotokollilla. Tietoliikenneprotokollat ovat kokoelma sääntöjä, jotka määrittävät, miten laitteet ja ohjelmat kommunikoivat sekä mahdolliset virheenpalautusmenetelmät. (13; 18.)

4.2 Verkkoratkaisujen esittely

Projektityötä varten piti löytää verkkoratkaisu, joka tukee asiakas-isännöityä mallia ja pystyy hyödyntämään Steamworks-rajapintaa. Verkkoratkaisut, joita tutkittiin olivat Mirror, Photon PUN ja Quantum, Fish-Net ja Networking For GameObjects.

Mirror

Mirror on ilmainen verkkokirjasto, joka on kopio (Fork) Unityn käytöstä poistetusta verkkoratkaisusta UNet (Unity Networking) (19; 20; 21; 22). Mirror on avoimen lähdekoodin (Open source) verkkoratkaisu eli sen lähdekoodia voi lukea ja muokata tarvittaessa (21; 22). Mirror sisältää sisäänrakennettuja sekä kolmannen osapuolen ylläpitämiä kuljetuskerroksia, joita voi vaihtaa tarpeen mukaan. Huomioitavana olisi FizzySteamworks-kuljetuskerros, jonka avulla voidaan yhdistää pelaajia Steamworks-rajapinnan välityspalvelimen kautta. (23.)

Photon PUN ja Quantum

PUN ja Quantum ovat Photonin-moninpelipalvelun tarjoamia verkkoratkaisuja (24). PUN (Photon Unity Networking) on Photonin vanha verkkoratkaisu. PUN sisältää työkaluja moninpelin tekemiseen, asiakas-palvelin-arkkitehtuurin ja pilvipalvelun, jonka avulla pelit isännöidään. (25.) PUN-verkkoratkaisun pilvipalvelussa on ilmainen kehitysversio ja maksullinen versio pelin julkaisua varten. Ilmaisessa kehitysversiossa voi olla 20 samanaikaista pelaajaa. Maksullinen julkaisuversio suurentaa sallittujen samanaikaisten pelaajien määrää ja sisältää kaiken, mitä on kehitysversiossa. (26.)

Photon Quantum on deterministinen ECS-kehys (Entity Component System) moninpeleille (27; 28). ECS on ohjelmistoarkkitehtuurimalli, jossa kaikki pelissä olevat asiat (ympäristö, pelaajat, objektit jne.) ovat kokonaisuuksia (containers), joihin lisätään komponentteja (sijainti, nopeus jne.), ja pelin järjestelmät tekevät muutoksia näihin kokonaisuuksiin niissä olevien komponenttien mukaan. ECS-ohjelmistoarkkitehtuurimalli toimii toisin, kuin OOP (Object Oriented Programming), jossa objektit ovat vastuussa itsensä ja ympärillään olevien objektien muuttamisesta. (29.)

ECS-kehyksessä pelaajien syötteet lähetetään palvelimelle, joka päättää, onko syöte kelvollinen. Asiakaslaitteet suorittavat omaa simulaatiotansa, joka ennustaa muiden pelaajien syötteitä, ja virheellisen ennusteen tapahtuessa Quantumin rollback-systeemi palauttaa pelitilanteen ja simuloi sen uudelleen Photon-palvelimen vahvistamilla syötteillä. (27; 28.) Quantum-verkkoratkaisun pilvipalvelua voi käyttää pelikehityksessä ja julkaisussa ilmaiseksi. Pelikehitys-versiossa voi olla 20 pelaajaa samanaikaisesti ja julkaisu-versiossa voi olla 100 pelaajaa samanaikaisesti. Maksullisilla paketeilla voi suurentaa sallittua samanaikaisten pelaajien määrää. (30.)

Pelaajien yhdistäminen peliin näiden Photon-verkkoratkaisujen avulla toimii Photon Realtime-verkkoinfrastruktuurin kautta, jonka avulla palvelin voi liittää pelaajan joko satunnaiseen peliin tai hakea listan peleistä, joista pelaaja voi valita mihin liittyä. Pelaajat voivat myös luoda oman pelihuoneen ja asettaa sille

yksilöllisen nimen, jolloin pelaajat, jotka tietävät huoneen nimen, voivat liittyä peliin palvelimen kautta. (31; 32.) Photon-verkkoratkaisujen käyttäminen edellyttää Photon-pilvipalvelun käyttämistä.

Fish-Net

Fish-Net on ilmainen verkkokirjasto Unity-pelimoottorille. Fish-Net ei perustu aiemmin rakennettuihin ratkaisuihin. Fish-Net on palvelin-valtuutettu eli se sallii omien palvelimien käytön ja antaa asiakkaiden toimia palvelimena ja asiakkaana. (33.) Fish-Net-verkkoratkaisulle on saatavilla kuljetuskerroksia, joita voi vaihtaa tarpeen mukaan. Tätä työtä varten huomioidaan FishySteamworks, jonka avulla voi yhdistää pelaajia Steamworks-rajapinnan avulla. (34.)

Netcode for GameObjects

Netcode for GameObjects (NGO) on Unityn oma ilmainen avoimen lähdekoodin verkkokirjasto (35). Verkkokirjasto sisältää Unityn oman Unity Transport-kuljetuskerroksen, mutta yhteisön tekemiä kuljetuskerroksia voi myös käyttää. Huomioitavana on yhteisön tekemä SteamNetworkingSockets-kuljetuskerros, jonka avulla voi hyödyntää Steamworks-rajapintaa pelaajien yhdistämiseen ja tiedon välittämiseen. (18.) Unity tarjoaa oman välityspalvelimen vertaisverkko-pelien luomista varten. Tämä välittäjäpalvelin on ilmainen tiettyyn käyttömäärään asti, jonka jälkeen maksu menee PAYG (Pay as you go) maksusuunnitelman mukaan. (36.)

4.3 Verkkoratkaisun valinta

Projektityön tekemiseen valittiin Mirror-verkkokirjasto. Valinta perustuu asiakas-isäntä-verkkomallin tukemiseen, olemassa olevaan dokumentaatioon, aiheeseen liittyviin verkkoaineistoihin ja testaamisen onnistumiseen. Mirror-verkkoratkaisua testattiin erillisessä projektissa, jossa tarkistettiin, pystyykö asiakaslaite isännöimään pelin ja pystyykö Steam-alustan kautta liittymään tähän isännöityyn peliin.

5 Verkkopelimuodon toteutus

Projektityön verkkopelimuodon toteuttaminen tehtiin verkkomallin mukaisesti hyödyntämällä Mirror-verkkoratkaisun dokumentaatiota, verkosta löydettyjä verkkoaineistoja ja videomateriaaleja, joita sovellettiin tarpeiden mukaan. Tässä luvussa vanhaa pelimuotoa kutsutaan paikallispelimuodoksi ja uutta verkon kautta toimivaa pelimuotoa kutsutaan verkkopelimuodoksi.

5.1 Toiminnallisuuden muutokset

Projektityön tekeminen aloitettiin luomalla uudet aulat moninpelimuodoille päävalikkoon. Uusiin auloihin siirrytään niille tarkoitetuilla napeilla päävalikossa (kuva 13). Local-nappi vie paikallispelimuodon aulaan ja Online-nappi vie verkkopelimuodon aulavalikkoon.

Kuva 13. Uusi päävalikko (2).

Paikallispelimuodon aula toimii samalla tavalla kuin ennen, mutta siihen lisättiin paneeli aulan ja pelaajien nimille (kuva 14).

Kuva 14. Uusi paikallispelimuodon aula (2).

Verkkopelimuodon aulaan siirrytään aulavalikon (kuva 15) kautta, kun pelaaja joko isännöi pelin tai kun pelaaja liittyy toisen pelaajan isännöimään peliin.

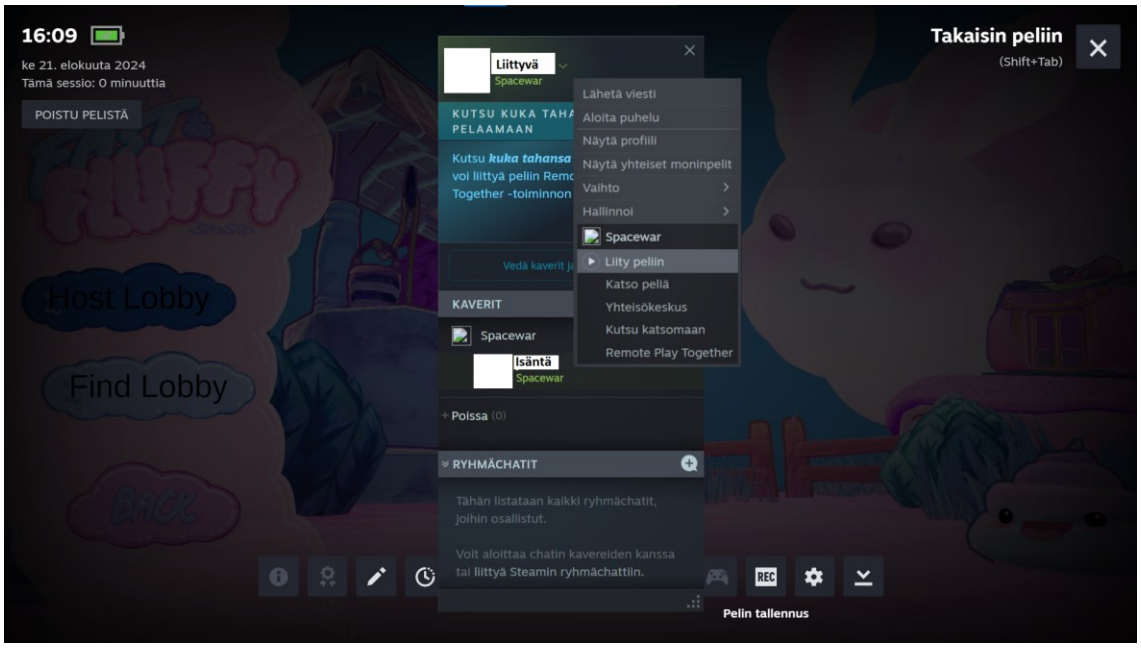
Kuva 15. Verkkopelimuodon aulavalikko (2).

Aulan luonnin ja liittymisen toiminta liitettiin aulavalikossa Host Lobby- ja Join Lobby -nappeihin. Pelaajat pystyvät isännöimään aulan Host Lobby -napilla ja avaamaan aulalistan Find Lobby -napilla. Verkkopelimuodon aula (kuva 16) toimii samalla tavalla kuin paikallinen aula sillä erolla, että pelaajat liittyvät verkon

kautta Steamworks-rajapinnan avulla. Kun isäntä luo aulan, hänelle luodaan pelaajahahmo aulan 3D-tilaan ja hänen Steam-käyttäjänimensä ja profiilikuvansa asetetaan nimilistaan (kuva 16).

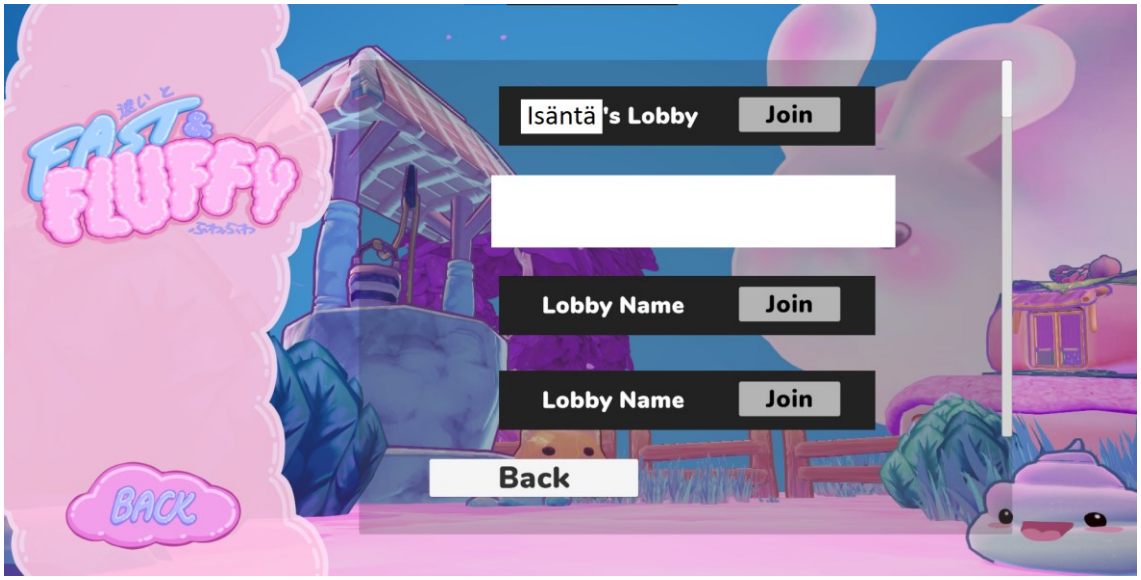
Kuva 16. Verkkopelimuodon aula (2).

Pelaajat pystyvät liittymään aulaan joko Steam-ystävälistan kautta (kuva 17) tai aulalistan kautta (kuva 18), josta voi liittyä listattuihin auloihin. Steam-ystävälistan kautta voi liittyä joko valitsemalla kaverin, joka on isännöi aulaa, ja liittymällä suoraan, tai isäntä voi kutsua liittymään.



Kuva 17. Steam-ystävälista-paneeli (2).

Aulalistan (kuva 18) avulla voi liittyä toisen pelaajan isännöityyn aulaan pelin sisällä. Aulalista listaa kaikki julkiset aulat, jotka löytyvät Steamworks-rajapinnasta.



Kuva 18. Aulavalikon aulalista (2).

Pelaajien liittyessä aulaan heille luodaan pelaajahahmot ja heidän Steam-käyttäjänimensä ja profiilikuvat asetetaan näkyviin aulassa olevaan nimilistaan (kuva 19). Isäntä-pelaaja voi aloittaa pelitasoon siirtymisen painamalla aulassa olevaa Start Game -nappia. Vain isännöivä pelaaja voi aloittaa pelitasoon siirtymisen.

Kuva 19. Verkkopelimuodon aula (2).

Ennen kuin pelin voi aloittaa, kaikkien pelaajien pitää ladata pelitaso, jotta pelitason alkuanimaatio ja pelinaloitusajastin voidaan aktivoida. Pelitason toiminta on pysäytetty, kunnes kaikki pelaajat ovat ladanneet pelitason. Pelin aikana pelitasosta poistuminen toimii taukovalikon (kuva 20) kautta, josta pelin isäntä voi palauttaa kaikki pelaajat takaisin aulaan tai asiakaspelaaja voi lähteä pelistä.

Kuva 20. Taukovalikko verkkopelimuodossa (2).

Asiakas voi poistua pelistä taukovalikon kautta Main Menu -napilla. Asiakkaan poistuessa pelistä hänet asetetaan takaisin päävalikkoon. Pelitasoa ei pysäytetä taukovalikon auki olemisen aikana verkkomoninpelissä, jotta sitä ei pysty käyttämään muiden pelaajien häiritsemiseen. Pelitason toiminta muuten toimii samalla tavalla paikallispelimuodossa, eli pelaajat pystyvät liikkumaan, sivuluisumaan ja käyttämään esineitä.

5.2 NetworkManager-komponentin alustaminen

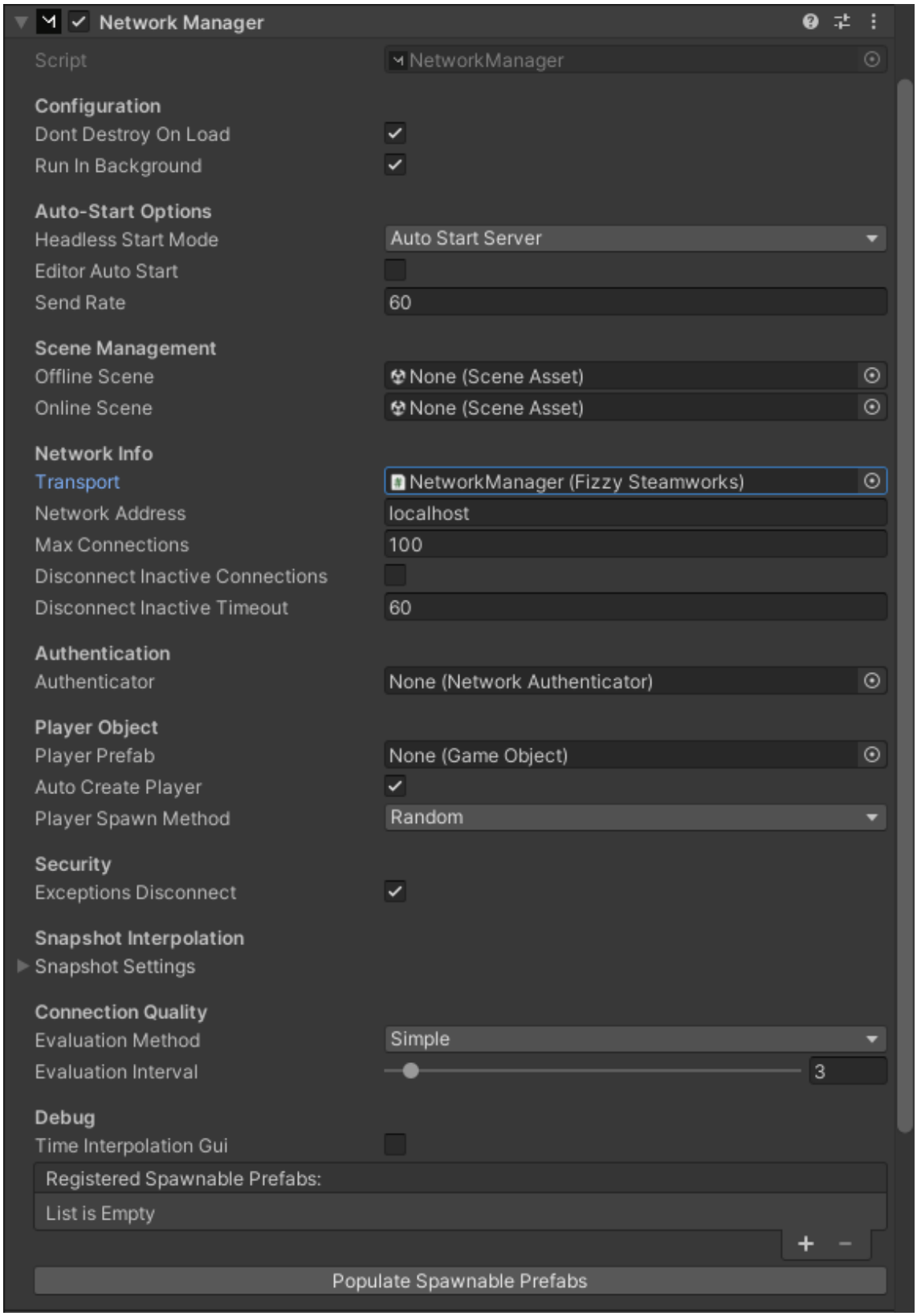
Verkkotoiminnallisuuden toteuttaminen tehtiin hyödyntämällä Mirror-verkkoratkaisua ja sen FizzySteamworks-kuljetuskerrosta. Mirror-verkkoratkaisu hankittiin ensin Unity Asset kaupasta ja tuotiin projektiin (37). FizzySteamworks-kuljetuskerros (versio 4.4.1) ladattiin GitHubista Unity-paketti-tiedostona ja tuotiin projektiin. Tämä paketti sisältää Steamworks.NET C# -kääreen (wrapper), jota kuljetuskerros tarvitsee Steamworks-rajapinnan kutsumiseen. (38.)

Verkkotoiminnallisuuden luominen aloitettiin luomalla NetworkManager-objekti, jonka tehtävänä on hallita verkkomoninpeliin liittyviä keskeisiä toimintoja. Tähän

objektiin lisättiin CustomNetworkManager-, FizzySteamworks-, SteamManager- ja SteamLobby-komponentit.

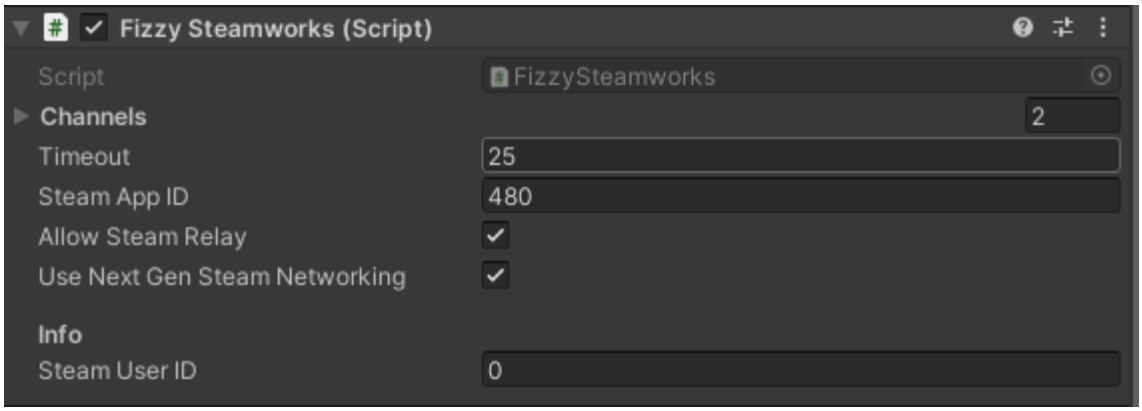
CustomNetworkManager-komponentti on skripti, joka perii Mirror-verkkoratkaisun NetworkManager-komponentin, eli se sisältää kaiken, mitä NetworkManager-komponentti sisältää, mutta sitä voi räätälöidä tarpeiden mukaan.

NetworkManager-komponentti (kuva 22) hoitaa pelin yleisten verkkotoimintojen hallintaa. Esimerkiksi huolehtii siitä, että peli suoriutuu joko asiakas-, palvelin- tai isäntätilassa ja että pelin objektit (pelaajahahmot, esineet jne.) luodaan niin, että ne ovat synkronisoitu kaikille pelaajille ja että aktiivisen scene-tiedoston vaihtaminen suoriutuu jokaiselle pelaajalle. (39.)



Kuva 21. NetworkManager-komponentti (2).

FizzySteamworks-komponentti (kuva 23) on kuljetuskerros, jonka avulla pystytään lähettämään pelitietoja isännän ja pelaajien laitteiden välillä Steamworks-välittäjäpalvelimen kautta. Tämä kuljetuskerros toimii hyödyntämällä Steamworks-rajapintaa (Steamworks.NET:n kautta) yhdistämällä tai välittämällä yhteyden muihin pelaajiin. (40.) NetworkManager-komponentti tarvitsee FizzySteamworks-komponentin referenssin, jotta se voi lähettää pelitietoa verkon kautta pelaajien välillä.



Kuva 22. FizzySteamworks-komponentti (2).

SteamManager-komponentti käsittelee Steamin ja pelin välisen yhteyden käynnistämisen ja sammutuksen. SteamManager-komponentti tarvitaan, jotta voidaan kutsua Steamworks-rajapinnasta funktioita. (41.) SteamLobby-komponentti on skripti, joka luotiin aulaan liittyvien toimintojen tekemiseen. SteamLobby-skriptin toimintaa avataan syvemmin luvussa 5.3.

5.3 Isännöinnin toteuttaminen

Isännöinti ja liittyminen toteutettiin SteamLobby-skriptin avulla, joka hoitaa aulan isännöimistä, julkisten aulojen hakemista ja niihin liittymistä. SteamLobby-skripti toimii kutsumalla SteamMatchmaking-skriptissä olevia funktioita, joiden avulla pystytään luomaan auloja, hakemaan olemassa olevia auloja ja liittymään määriteltyyn aulaan. SteamMatchmaking-skripti kuluu Steamworks-rajapinnan sisältöön, jota pystytään hyödyntämään Steamworks.NET:n avulla, joka tuli FizzySteamworks-paketin mukana (42).

Isännöinti tapahtuu kutsumalla SteamMatchmaking-skriptissä olevaa CreateLobby-funktiota, joka ottaa parametrina aulatyypin (julkinen, yksityinen tai vain ystävät) ja yhteyksien sallitun maksimimäärän (42). Esimerkkikoodissa 1 näkyy, miten CreateLobby-funktiota käytetään koodissa. CreateLobby-funktio kutsuu Steamworks-rajapintaa luomaan uuden julkisen aulan ja asettaa sille pelaajien maksimimäärän.

```
public void HostLobby()
{
    SteamMatchmaking.CreateLobby(ELobbyType.k_ELobbyTypePublic,
        networkManager.maxConnections);
}
```

Esimerkkikoodi 1. Funktio aulan isännöimiseen.

CreateLobby-funktion tuloksen tiedon palauttaa LobbyCreated_t-takaisinkutsu, joka kertoo, onnistuiko aulan luominen ja onko isäntä-pelaaja luodussa aulassa. Aulan luonnin onnistuessa LobbyCreated_t-takaisinkutsu palauttaa luodun aulan SteamID:n, jota käytetään aulan tietojen asettamiseen, jotta muut pelaajat pystyvät liittymään. (42.) Esimerkkikoodissa 2 näkyy LobbyCreated_t-takaisinkutsu koodissa. LobbyCreated_t-takaisinkutsu asettaa aulan tiedot ja kutsuu NetworkManager-komponenttia asettamaan laiteen toimimaan isäntä-tilassa.

```
private void OnLobbyCreated(LobbyCreated_t callback)
{
    // Check if lobby has been created
    if(callback.m_eResult != EResult.k_EResultOK) { return; }

    networkManager.StartHost();

    // Set lobby data
    CSteamID steamID = new CSteamID(callback.m_ulSteamIDLobby);
    SteamLobbyID = steamID;

    SteamMatchmaking.SetLobbyData(steamID, hostKey,
        SteamUser.GetSteamID().ToString());

    SteamMatchmaking.SetLobbyData(steamID, "name",
        SteamFriends.GetPersonaName().ToString() + "'s Lobby");
}
```

Esimerkkikoodi 2. Takaisinkutsu aulan luomisesta.

Aulan tietoihin asetetaan isäntä-pelaajan SteamID ja käyttäjänimi. Käyttäjänimi käytetään aulan nimen asettamiseen ja isännän SteamID tarvitaan, jotta pystytään yhdistämään liittyvät pelaajat Steamworks-rajapinnan kautta isäntään.

5.4 Liittymisen toteuttaminen

Liittyminen pelissä toimii joko aulalistan tai Steam-ystävälistan kautta. Steam-ystävälistan kautta liittyminen toimii Steam Overlayn avulla. Steam Overlay on osa Steam-alustan käyttäjäliittymää, joka toimii pelin taustalla. Pelissä Steam Overlaytä käytetään pelaajien liittymisen suorittamiseen ystävälistan kautta. (39.) Steam Overlay aktivoituu ja kytkeytyy peliin automaattisesti, jos peli avataan Steam-alustan kautta. Kehityksessä olevissa peleissä esimerkiksi Unity-pelimoottorissa Steam Overlay aktivoituu, kun kutsutaan SteamAPI_Init-funktiota, jonka Steamworks.NET hoitaa valmiiksi. (43.)

Aulalistan kautta liittymisen toiminta saatiin tehtyä hakemalla Steamworks-rajapinnasta lista julkisia auloja. Esimerkkikoodissa 3 näkyy, kuinka Steamworks-

rajapinnasta haetaan lista auloja. Aulalistan pyyntöä ennen asetetaan suodatin, joka rajoittaa tulosten määrää, minkä jälkeen kutsutaan SteamMatchmaking-skriptistä RequestLobbyList-funktiota aulalistan saamiseksi.

```
public void GetLobbyList()
{
    if (lobbyIDs.Count > 0) { lobbyIDs.Clear(); }

    SteamMatchmaking.AddRequestLobbyListResultCountFilter(50);
    SteamMatchmaking.RequestLobbyList();
}
```

Esimerkkikoodi 3. Funktio aulalistan hakemiseen Steamworks-rajapinnasta.

RequestLobbyList-funktion onnistumisessa saadaan LobbyMatchList_t-takaisinkutsu, josta saadaan haussa saatujen aulojen määrä (42). Esimerkkikoodissa 4 käytetään LobbyMatchList_t-takaisinkutsua lisäämään Steamworks-rajapinnasta haettujen aulojen SteamID-arvot koodissa olevaan listaan. Tässä funktiossa kutsutaan lisäksi SteamMatchmaking-skriptistä RequestLobbyData-funktiota, joka palauttaa aulojen tiedot.

```
public void OnGetLobbyList(LobbyMatchList_t result)
{
    if (ServerPanelManager.Instance.lobbyList.Count > 0)
          { ServerPanelManager.Instance.DestroyLobbies(); }

    for(int i = 0; i < result.m_nLobbiesMatching; i++)
    {
        CSteamID lobbyId = SteamMatchmaking.GetLobbyByIndex(i);
        lobbyIDs.Add(lobbyId);
        SteamMatchmaking.RequestLobbyData(lobbyId);
    }
}
```

Esimerkkikoodi 4. Takaisinkutsu aulalistan vastaanottamiseen.

RequestLobbyData-funktion onnistuessa saadaan LobbyDataUpdate_t-takaisinkutsu, josta saadaan haetun aulan tiedot (42). Esimerkkikoodissa 5 käytetään LobbyDataUpdate_t-takaisinkutsua kutsumaan ServerPanelManager-skriptiä asettamaan aulojen tiedot aulalistan käyttöliittymään haettujen aulatietojen avulla ja listalla, johon kerättiin aulojen SteamID:t.

```
public void OnGetLobbyData (LobbyDataUpdate_t result)
{
    ServerPanelManager.Instance.DisplayLobbies(lobbyIDs, result);
}
```

Esimerkkikoodi 5. Takaisinkutsu aulojen tietojen vastaanottamiseen.

Lopulta ServerPanelManager tarkistaa, että aulojen tiedot ovat oikein ja luo jokaiselle aulalle objektin ServerPanelLobbyData, johon lisätään aulojen tiedot, jotka asetetaan aulalistaan näkyviin. Nämä objektit sisältävät tiedon aulan nimestä, aulan SteamID:n ja funktion, joka kutsuu JoinLobby-funktiota SteamLobby-skriptissä objektissa olevan aulan tiedoilla. Esimerkkikoodissa 6 näkyy JoinLobby-funktio SteamLobby-skriptissä. JoinLobby-funktio ottaa parametrinä halutun aulan SteamID:n ja yhdistää pelaajan Steamworks-rajapinnan kautta haluttuun aulaan.

```
public void JoinLobby(CSteamID lobbyId)
{
    SteamMatchmaking.JoinLobby(lobbyId);
}
```

Esimerkkikoodi 6. Funktio aulaan liittymiseen.

JoinLobby-funktion onnistumisesta ilmoittaa LobbyEnter_t-takaisinkutsu. Tämä takaisinkutsu antaa parametrinä aulan metadatan käyttöön (42). Esimerkkikoodissa 7 LobbyEnter_t-takaisinkutsussa asetetaan aulan tiedot ja NetworkManager-komponenttia kutsutaan asettamaan laite toimimaan asiakas-tilassa. NetworkManager-komponentille lisäksi asetetaan networkAddress-muuttujaan aulan SteamID-arvon, jonka avulla se pystyy yhdistämään asiakkaan isäntään, kun StartClient-funktiota kutsutaan.

```
private void OnLobbyEnter(LobbyEnter_t callback)
{
    CurrentLobbyID = callback.m_ulSteamIDLobby;
    CSteamID steamID = new CSteamID(callback.m_ulSteamIDLobby);

    //Clients
    if (NetworkServer.active) { return; }

    networkManager.networkAddress = SteamMatchmaking.GetLobbyData(steamID, host-
Key);
    networkManager.StartClient();
}
```

Esimerkkikoodi 7. Takaisinkutsu, joka kutsutaan aulaan liittymisen jälkeen.

Pelaajien liittyessä aulaan heille luodaan Online-Player-objekti, joka edustaa pelaajaa verkkopelimuodossa. Online-Player-objekti selitetään syvemmin seuraavassa luvussa (5.5).

5.5 Synkronisointi

Pelissä synkronoidaan skenen vaihtaminen, pelaajahahmot ja esineet verkon kautta kaikille pelaajille. Synkronisointi toteutettiin hyödyntämällä Mirror-verkkoratkaisussa olevia komponentteja, verkkoattribuutteja ja funktioita.

Skenen vaihto moninpelissä päävalikon ja pelitason välillä tehdään CustomNetworkManager-komponentin avulla, joka pystyy vaihtamaan skenen määriteltyyn skeneen jokaiselle liittyneelle pelaajalle. Pelitasoon siirtyminen verkkoaulasta tapahtuu Start Game -napilla, joka on asetettu kutsumaan CustomNetworkManager-skriptiä vaihtamaan aktiivinen skene määriteltyyn skeneen kaikille asiakaslaitteille. Skenen vaihdon tekee funktio ServerChangeScene, joka ottaa parametrinä määritellyn skenen nimen merkkijono-muuttujana.

Pelaajahahmon synkronointi tehtiin luomalla erillinen kopio pelaajahahmo-objektista Online-Player, johon lisättiin komponentit NetworkIdentity, NetworkTransform, NetworkAnimator ja NetworkRigidbody. Nämä komponentit mahdollistavat pelaajahahmon luomisen ja tunnistamisen verkkopelissä sekä sijainnin, animaatioiden ja RigidBody-komponentin synkronisoinnin.

Pelaajien liikkuminen tapahtuu asiakaslaitteella samalla tavalla kuin paikallispelimuodossa Input-systeemin avulla, joka on Unityn oma paketti pelaajien syötteiden tulkitsemiseen (44). NetworkTransform-komponentti synkronisoi pelaajanhahmon sijainnin asiakaslaitteelta kaikille pelaajille.

Online-Player-objektiin luotiin myös OnlinePlayerController ja OnlinePlayerPowerUpHandler-komponentit. OnlinePlayerController-komponentti on skripti, joka edustaa pelaajaa verkkopelimuodossa. OnlinePlayerController-komponentti sisältää pelaajaan liittyvää tietoa (nimi ja ID:t) ja pelaajan hallintaan liittyviä funktioita. OnlinePlayerPowerUpHandler-komponentti on skripti, joka perii alkuperäisen PlayerPowerUpHandler-skriptin, jotta esineen käyttäminen voidaan räätälöidä verkkopelimuotoa varten.

Verkkopelimuodossa esineiden käytön tieto pitää välittää kaikille pelaajille, jotta se toimii ja näkyy oikein. Tämä saatiin toteutettua Mirror-verkkoratkaisussa olevien verkkoattribuuttien avulla, jotka määrittelevät koodissa funktion suoritettavaksi, joko asiakas- tai isäntälaitteella (45). Projektityössä käytetään enimmäkseen Command- ja ClientRpc-attribuutteja.

Command-attribuutti merkitsee funktion suoritettavaksi palvelimella tai isäntälaitteella, kun sitä kutsutaan asiakaslaitteelta. Esimerkiksi kun asiakaslaite kutsuu tätä merkittyä funktiota, palvelin suorittaa sen. (45.) ClientRpc-attribuutti merkitsee funktion suoritettavaksi kaikille asiakaslaitteille, kun palvelin kutsuu sitä. Esimerkiksi kun palvelin kutsuu tätä merkittyä funktiota, jokainen asiakaslaite suorittaa tämän funktion. (45.)

Näiden attribuuttien avulla toteutetaan esineen asettaminen ja käyttäminen pelaajille. Ensin asiakaslaite ilmoittaa Command-attribuutin avulla, että isäntälaite tekee toiminnon, ja isäntälaite välittää ClientRpc-attribuutilla tiedon kaikille pelaajille.

Attribuutteja pystyy käyttämään koodissa lisäämällä ennen funktiota attribuuttimerkinnän muodossa [”Attribuutti-nimi”] esimerkiksi [Command] tai [ClientRpc]. Esimerkkikoodissa 8 näkyy, miten koodissa käytetään verkkoattribuutteja esineiden käytön välittämiseksi verkon kautta kaikille pelaajille. Koodissa asiakaslaite kutsuu esineen käytössä Command-attribuutilla merkittyä funktioita CmdUsePowerUpDirection, jossa kutsutaan ClientRpc-attribuutilla merkittyä funktiota RpcUsePowerUpDir, joka lopulta käyttää esineen, jokaisella asiakaslaitteella.

```
[Command]
public void CmdUsePowerUpDirection(int dir)
{
    RpcUsePowerUpDir(dir);
}

[ClientRpc]
public void RpcUsePowerUpDir(int dir)
{
    PlayerUsePowerUp(dir);
}
```

Esimerkkikoodi 8. Esimerkki Command- ja ClientRpc-attribuuttien käytöstä koodissa.

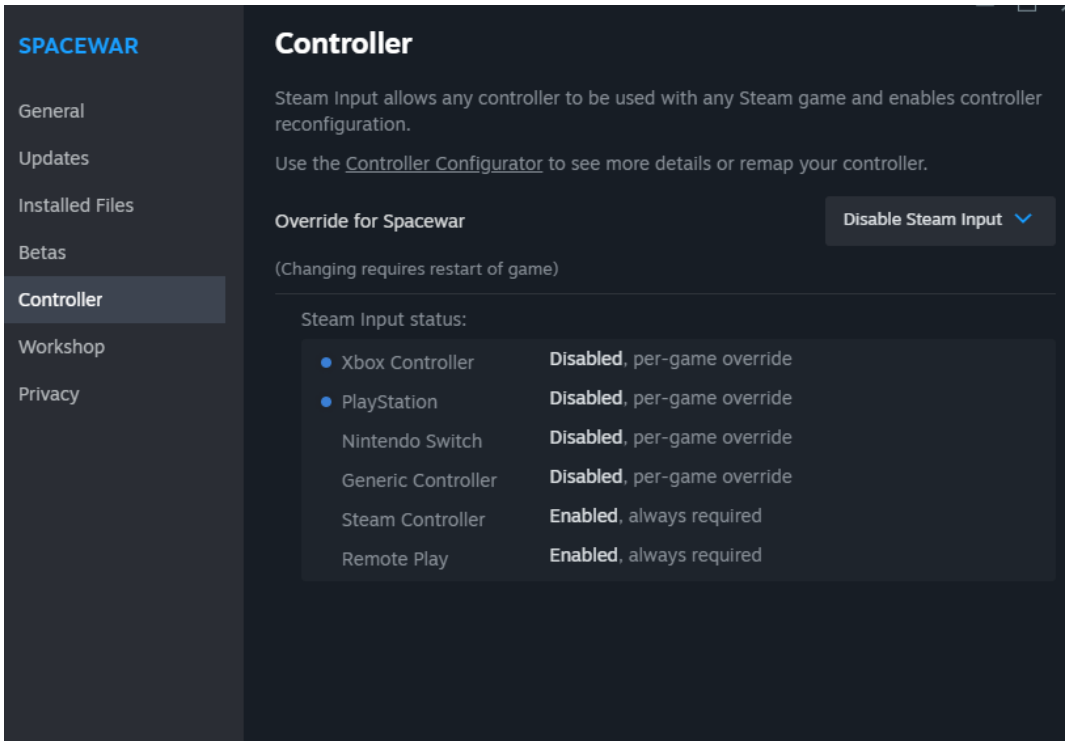
Esineiden asettaminen pelaajalle synkronisointiin kaikille pelaajille myös Command- ja ClientRpc-attribuuttien avulla.

Esineisiin lisättiin NetworkIdentity- ja NetworkAnimator-komponentit, jotta ne toimisivat verkkomoninpelin aikana ja niiden animaatiot näkyisivät kaikille pelaajille. Online-Player-objekti ja esine-objektit rekisteröitiin CustomNetworkManager-komponenttiin. Online-Player-objekti asetettiin CustomNetworkManager-komponentissa pelaajaobjektiksi, ja esineet lisättiin Spawnable Prefabs -osioon, jotta niitä voisi luoda pelin aikana. CustomNetworkManager-komponentti lisäksi luo Online-Player-objektit, kun aula luodaan tai pelaajat liittyvät.

5.6 Haasteet

Projektityön toteuttamisen aikana tuli vastaan haasteita erityisesti synkronoinnissa, ohjaimien yhdistämisessä ja pelin testaamisessa. Pelaajien ilmestymisen synkronointi pelitasoon aiheutti ongelmia, koska pelaajien pitäisi liittyä samaan aikaan niin, että pelin alkuanimaatio ja aloitusajastin toimisivat oikein. Tämän saavuttamiseksi pysäytettiin pelin toiminta, kunnes kaikki pelaajat olivat siirtyneet pelitasoon. Pelaajien esineiden synkronoinnissa tuli ongelma, kun pelaaja käytti esineen, jota ei löytynyt tai joka ei ollut samanlainen isäntälaitteen pelisimulaatiossa. Tässä tapauksessa pelaajat menettivät yhteyden isäntään. Tämä korjattiin varmistamalla, että pelaajan haltuun tulevat esineet luodaan myös isäntälaitteen simulaatiossa pelaajalle.

Pelaajien ohjaimien yhdistämisessä tuli ongelmia, kun laite ei pystynyt yhdistämään ohjainta peliin. Tämän ongelman aiheutti Steam-ohjainasetukset, jossa piti asettaa Steam Input -ominaisuus pois päältä Override-asetuksessa. (kuva 24).



Kuva 23. Steam ohjainasetukset.

Testaamisessa tuli ongelmia, kun testattiin kahden pelaajan toimintaa pelissä verkon kautta. Testaamisessa pelin suorittamiseen tarvittava .exe-tiedosto ja datakansio piti siirtää toiselle laitteelle. Toisella laitteella tiedostot piti myös lisätä toisen Steam-käyttäjän pelikirjastoon, jotta peliä voisi pelata Steam-alustalla. Tämä oli erittäin aikaa vievää, varsinkin kun tuli virheitä, jotka tapahtuivat, kun pelaajia oli enemmän, kuin yksi.

5.7 Lopputulos

Lopputuloksena saatiin aikaan toimiva verkkopelimuoto, johon pelaajat pystyvät liittymään Steam-ystävälistan tai pelin sisäisen aulalistan kautta. Peliin luotiin erilliset aulat paikallis- ja verkkopelimuodoille. Pelaajien liittyessä verkkopelimuodon aulaan heille luodaan pelaajahahmo ja heidän Steam-käyttäjänimensä ja profiilikuvansa asetetaan nimilistaan.

Pelin isäntä pystyy aloittamaan pelin, joka siirtää kaikki pelaajat pelitasoon. Pelaajien sijainti ja esineiden käyttö on synkronoitu kaikille muille pelaajille verkon kautta. (kuva 25).

Kuva 24. Liittyneen pelaajan pelinäkymä (2).

Liittyneet pelaajat pystyvät lähtemään isännöidystä pelistä, joka poistaa pelaajan pelaajahahmon. Jos isäntä lähtee pelitasosta, kaikki pelaajat palautetaan aulaan, ja jos isäntä poistuu aulasta, se suljetaan, eli kaikki pelaajat poistetaan aulasta. Peli toimii muuten samalla tavalla kuin paikallisessa pelimuodossa.

6 Yhteenveto

Projektityön tavoitteena oli toteuttaa verkkopelimuodon lisääminen olemassa olevaan peliprojektiin, jotta useampi pelaaja pystyisi pelaamaan yhdessä verkon kautta Steamworks-rajapinnan avulla. Projektityön alkutilanteessa oli jo valmiiksi luotu kilpailupeli, johon paikallispelimuoto oli jo toteutettu.

Ensin tutkittiin, millä verkkomallilla voisi toteuttaa verkkopelimuodon, ja mikä verkkoratkaisu mahdollistaisi tämän verkkomallin toiminnan Steamworks-rajapinnan kautta. Verkkomalleista tutkittiin asiakas-palvelin-, vertaisverkko- ja asiakas-isäntä-mallit. Projektityön toteuttamista varten valittiin asiakas-isäntä-verkkomalli, jossa yksi asiakaslaite toimii isäntänä eli pelaajana ja palvelimena. Isännän laite suorittaa pelisimulaatiota, joka on pelitilanteen lopullinen päättäjä, ja tähän isäntään kaikki muut pelaajat liittyvät verkon kautta.

Projektityö lopulta toteutettiin Mirror-verkkoratkaisun ja sen FizzySteamworks-kuljetuskerroksen avulla. Mirror-verkkoratkaisun avulla lisättiin verkkopelimuodon toimintaa, kuten pelaajien ja esineiden synkronisointia, pelitilanteen hallintaa ja skenen vaihtamista. FizzySteamworks-kuljetuskerros mahdollisti pelaajien yhdistämisen peliin ja pelitietojen lähettämisen isännän ja muiden pelaajien laitteiden välillä Steamworks-välittäjäpalvelimen kautta. Peliin luotiin myös uudet erilliset aulat paikallis- ja verkkopelimuodille.

Lopputuloksena oli toimiva verkkopelimuoto, jossa pystytään isännöimään aula tai liittymään olemassa olevaan aulaan. Pelaajien ja esineiden toiminta on synkronoitu verkon kautta kaikille pelaajille Steamworks-rajapinnan avulla. Projektityön toteutus täytti asetetut tavoitteet, mutta joitakin asioita olisi voinut tehdä toisin, kuten löytää erilaisen tavan pelin testaamiseen. Projektityön toteuttamisesta saavutettiin parempi ymmärrys verkkopelimuotojen toiminnasta ja toteuttamisesta Unity-pelimoottorissa sekä Steam-alustan ja Steamworks-rajapinnan toiminnasta.

Lähteet

1 Scenes. Verkkoaineisto. Unity. <https://docs.unity3d.com/560/Documentation/Manual/CreatingScenes>. Luettu 6.5.2024.

2 Fast and Fluffy -pelisovellus. 2023. Rantala, Leevi ym.

3 An Introduction to multiplayer network and server models. Verkkoaineisto. Unity. <http://web.archive.org/web/20240522015822/https://unity.com/how-to/intro-network-server-models>. Luettu 5.1.2024.

4 Network topologies. Verkkoaineisto. Unity. <https://docs-multiplayer.unity3d.com/netcode/current/terms-concepts/network-topologies/>. Luettu 4.1.2024.

5 Maltbe, Nick. 2023. How Multiplayer Games Work. Verkkoaineisto. <https://www.youtube.com/watch?v=KBBJqPL5-eU>. 18.1.2023. Katsottu 4.1.2024.

6 Understanding Networking Types and their Benefits for each Game Types. Verkkoaineisto. Edgegap. <https://edgegap.com/blog-en/explainer-series-authoritative-servers-relays-peer-to-peer-understanding-networking-types-and-their-benefits-for-each-game-types>. Luettu 5.1.2024.

7 Listen server and host architecture. Verkkoaineisto. Unity. <https://docs-multiplayer.unity3d.com/netcode/1.1.0/learn/listen-server-host-architecture/>. Luettu 5.1.2024.

8 Relay servers. Verkkoaineisto. Unity. <https://docs.unity.com/ugs/manual/relay/manual/relay-servers>. Luettu 5.1.2024.

9 How To Forward a Port. 2022. Verkkoaineisto. Port Forward. <https://portforward.com/>. Päivitetty 17.11.2022. Luettu 14.4.2024.

10 Port Forwarding to a Game Server. 2024. Verkkoaineisto. Port Forward. <https://portforward.com/game-servers/>. Päivitetty 23.10.2024. Luettu 14.4.2024.

11 How do network routers provide security? Verkkoaineisto. Allied Telesis. <https://www.alliedtelesis.com/fi/en/foundations/how-do-network-routers-provide-security>. Luettu 15.4.2024.

12 Wherry, Jack. 2024. What is port forwarding and how safe is it? Verkkoaineisto. <https://cybernews.com/what-is-vpn/port-forwarding/>. Päivitetty 19.1.2024. Luettu 14.4.2024.

13 House, Brandi. 2023. How to choose the right netcode for your game. Verkkoaineisto. <https://unity.com/blog/games/how-to-choose-the-right-netcode-for-your-game>. 8.9.2020. Luettu 4.1.2024.

14 Communications. Verkkoaineisto. Mirror Networking. <https://mirror-networking.gitbook.io/docs/manual/guides/communications>. Luettu 11.1.2024.

15 Synchronization. Verkkoaineisto. Mirror Networking. <https://mirror-networking.gitbook.io/docs/manual/guides/synchronization>. Luettu 11.1.2024.

16 Authority. Verkkoaineisto. Mirror Networking. <https://mirror-networking.gitbook.io/docs/manual/guides/authority>. Luettu 11.4.2024.

17 General. Verkkoaineisto. Mirror Networking. <https://mirror-networking.gitbook.io/docs/manual/general>. Luettu 8.1.2024.

18 Transports. Verkkoaineisto. Unity. <https://docs-multiplayer.unity3d.com/netcode/current/advanced-topics/transports/>. Luettu 5.1.2024.

19 A Brief History of Mirror. Verkkoaineisto. Mirror Networking. <https://mirror-networking.gitbook.io/docs/trivia/a-history-of-mirror>. Luettu 21.5.2024.

20 Migration Guide. Verkkoaineisto. Mirror Networking. <https://mirror-networking.gitbook.io/docs/manual/general/migration-guide>. Luettu 21.5.2025.

21 Mirror Networking. Verkkoaineisto. Mirror Networking. <https://github.com/MirrorNetworking/Mirror>. Luettu 8.5.2024.

22 What is Fork? 2021. Verkkoaineisto. Webopedia. <https://www.webopedia.com/definitions/fork/>. Päivitetty 24.5.2021. Luettu 21.5.2024.

23 Transports. Verkkoaineisto. Mirror Networking. <https://mirror-networking.gitbook.io/docs/manual/transports>. Luettu 8.1.2024.

24 Photon. Verkkoaineisto. Photon. <https://www.photonengine.com/>. Luettu 6.1.2024.

25 Photon Pun. Verkkoaineisto. Photon. <https://www.photonengine.com/pun>. Luettu 6.1.2024.

26 Photon Pun Pricing. Verkkoaineisto. Photon. <https://www.photonengine.com/pun/pricing>. Luettu 6.1.2024.

27 Photon Quantum. Verkkoaineisto. Photon. <https://www.photonengine.com/quantum>. Luettu 7.1.2024.

28 Quantum 3 Intro. Verkkoaineisto. Photon. <https://doc.photonengine.com/quantum/current/quantum-intro>. Luettu 7.1.2024.

29 Blanker, Brandon. 2020. Versus Series #1 - ECS vs OOP. Verkkoaineisto. <http://web.archive.org/web/20240105142937/https:/flamendless.github.io/ecs-vs-oop/>. 16.8.2020. Luettu 23.5.2024.

30 Photon Quantum Pricing. Verkkoaineisto. Photon. <https://www.photonengine.com/quantum/pricing>. Luettu 7.1.2024.

31 Online Session. Verkkoaineisto. Photon. <https://doc.photonengine.com/quantum/current/manual/game-session/starting-session>. Luettu 7.1.2024

32 Matchmaking Guide. Verkkoaineisto. Photon. <https://doc.photonengine.com/realtime/current/lobby-and-matchmaking/matchmaking-and-lobby#play-with-your-friends>. Luettu 7.1.2024.

33 Introduction. Verkkoaineisto. Fish-networking. <https://fish-networking.gitbook.io/docs>. Luettu 9.1.2024.

34 Transports. Verkkoaineisto. Fish-networking. <https://fish-networking.gitbook.io/docs/manual/general/transports>. Luettu 9.1.2024.

35 About Netcode for GameObjects. Verkkoaineisto. Unity. <https://docs-multiplayer.unity3d.com/netcode/current/about/>. Luettu 6.2.2024.

36 UGS Pricing. Verkkoaineisto. Unity. <https://unity.com/products/gaming-services/pricing>. Luettu 6.2.2024.

37 Mirror Asset. Verkkoaineisto. Mirror Networking. <https://assetstore.unity.com/packages/tools/network/mirror-129321>. Luettu 8.1.2024.

38 Hoffmann, Marco. 2020. FizzySteamworks Releases. Verkkoaineisto. <https://github.com/Chykary/FizzySteamworks/releases>. Luettu 8.1.2024.

39 Network Manager. Verkkoaineisto. Mirror Networking. <https://mirror-networking.gitbook.io/docs/manual/components/network-manager>. Luettu 11.1.2024.

40 FizzySteamworks Transport. Verkkoaineisto. Mirror Networking. <https://mirror-networking.gitbook.io/docs/manual/transports/fizzysteamworks-transport>. Luettu 8.1.2024.

41 Labrecque, Riley. 2014. SteamManager. Verkkoaineisto. <https://steamworks.github.io/steammanager/>. 2014. Luettu 26.5.2024.

42 ISteamMatchmaking Interface. Verkkoaineisto. Valve. <https://partner.steamgames.com/doc/api/ISteamMatchmaking>. Luettu 23.5.2024.

43 Steam Overlay. Verkkoaineisto. Valve. <https://partner.steamgames.com/doc/features/overlay>. Luettu 23.5.2024.

44 Input System. Verkkoaineisto. Unity. <https://docs.unity3d.com/Packages/com.unity.inputsystem@1.7/manual/index>. Luettu 17.5.2024.

45 Attributes Transport. Verkkoaineisto. Mirror Networking. <https://mirror-networking.gitbook.io/docs/manual/guides/attributes>. Luettu 11.1.2024.