Tampereen ammattikorkeakoulu

Moninpelikehitys käyttäen Unity Unet -verkkotyökaluja

Timo Rintala

OPINNÄYTETYÖ
Huhtikuu 2019

Tietojenkäsittely
Pelituotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Pelituotanto

RINTALA, TIMO:
Moninpelikehitys käyttäen Unity Unet -verkkotyökaluja

Opinnäytetyö 41 sivua
Huhtikuu 2019

Opinnäytetyön tavoite oli vertailla eri verkkotyökalujen tarjontaa ja niiden hyöty- ja haittapuolia opiskeluprojektien näkökulmasta. Lisäksi tavoitteena oli kasata tiivis paketti Unity Unet -verkkotyökalun käytöstä ja siihen liittyvistä paikoittain hankalistakin konsepteista. Opinnäytetyön tarkoituksena oli koota pääosin pelituotannon opiskelijoille suunnattu kirjallinen opas, jonka avulla jokainen hieman Unityä osaava henkilö pystyy luomaan verkkopeliin tarvittavat toiminnallisuudet ilman suurempia ongelmia.

Työnkulku moninpeliä tehdessä on mutkikasta, joten oikean tyyppisen verkkotyökalun valinnalla on iso vaikutus projektin etenemisen kannalta. Eri työkalujen vertailun tuloksena päädyttiin siihen, että Unity Unet on kelvollisin työkalu opiskelijoiden nopeita projekteja varten. Unity Unet tarjoaa kehittäjälle paljon valmiita komponentteja ja palveluita. Ne nopeuttavat verkottamisen opiskelua ja säästävät paljon aikaa, kun kaikkea ei tarvitse tehdä itse alusta alkaen.

Opinnäytetyössä käytiin läpi kaikki olennaiset Unity Unetin toiminnallisuudet, joita verkkopelin tekeminen vaatii. Koodiesimerkit ja selitykset pyrittiin pitämään mahdollisimman yleispätevinä, jolloin niistä on hyötyä myös muita verkkotyökaluja käytettäessä. Useimmat verkkotyökalut pohjautuvat samoihin periaatteisiin, jolloin yhden työkalun osaaminen auttaa huomattavasti toisen työkalun opiskelussa. Työ kannattaa siis lukea läpi, vaikka ajatuksena olisikin käyttää jotain muuta verkkotyökalua kuin Unity Unetiä.

Asiasanat: unity, unet, pelinkehitys, verkkotyökalut

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Game development

RINTALA, TIMO:
Multiplayer Game Development Using Unity Unet Networking Tools

Bachelor's thesis 41 pages
April 2019

The purpose of this thesis was to study different networking tools for game development and find the most suitable candidate for student game projects. The thesis also includes a short manual of how to use the chosen networking tool, Unity Unet.

The material studied was mostly gathered from internet articles and documentation, as there are no suitable sources of information in books regarding this subject. Unity Unet is such a new invention that information overall was difficult to find.

As a result Unity Unet was found to be the best networking tool for absolute beginners who have never done anything with networking before. This is the reason why it was chosen as the tool for this thesis.

It is really the developer’s choice which networking tools work for their game the best. All are done by professionals and are functional, although some require more effort to learn than others. Overall, Unity Unet has many of built-in components and tools, which are easy to learn for a beginner and do not take unnecessary time.

Key words: unity, unet, game development, networking tools

SISÄLLYS

LYHENTEET JA TERMIT

scripti	koodattu komponentti, koodi
IDE	Integrated Development Environment, ohjelmointiympäristö
assetti	pelissä käytetty tiedosto
LLAPI	Low Level Application Programming Interface, matalan tason ohjelmointirajapinta
HLAPI	High Level Application Programming Interface, korkean tason ohjelmointirajapinta
serveri	palvelin, jolla peliä pyöritetään
dedicated serveri	palvelin, jolla peliä pyöritetään mutta ei pelata samaan aikaan
client	käyttäjä, käyttäjän tietokone
prefab	malli, pohja peliobjektista
SyncVar	Synchronized Variable, synkronoitu muuttuja

1 JOHDANTO

Tässä opinnäytetyössä keskitytään Unityn vuonna 2014 julkaisemaan Unetiin. Unet on Unityyn sisäänrakennettu verkkotyökalupaketti, jota käyttämällä on mahdollista luoda verkossa moninpelattavia pelejä. Moninpelattavien pelien teko on aikaisemmin ollut tuskallista, mutta Unity Unetin moninpelityökalut ovat uusi helppo tapa käyttää ammattilaisten verkkoteknologioita ja infrastruktuuria ilman suurempia päänvaivoja (Juhl E. 2014).

Opinnäytetyön päätavoitteena on kasata tiivis paketti Unity Unetin verkkotyökalujen käytöstä ja siihen liittyvistä hieman hankalistakin konsepteista. Toisena tavoitteena on vertailla Unityyn liitettävien moninpelien tekoon tarkoitettujen palveluiden tarjontaa ja niiden hyöty- sekä haittapuolia. Tarkoituksena on myös saada kirjallinen osio, jolla kuka tahansa hieman Unityn käyttöä osaava henkilö pystyy luomaan peliinsä yksinkertaisen moninpelin ilman suurempia vastoinkäymisiä.

Tämä opinnäytetyö on suunnattu erityisesti pelipuolen opiskelijoille tai muuten vain pelialasta kiinnostuneille. Opinnäytetyössä ei käydä juurikaan läpi Unityn peruskäyttöön tarvittavia taitoja, vaan keskitytään pääosin Unityn sisälle integroituihin verkkotyökaluihin. Tämän opinnäytetyön tarkoitus on luoda yksinkertaisia esimerkkejä verkkopelin rakentamisesta, joita tutkimalla käydään lävitse Unity Unetin peruskäyttöä ja komponentteja.

Työnkulku moninpeliä tehdessä on huomattavasti monimutkaisempaa, kuin yksinpeliä tehdessä. Moninpeleissä pelintekijän ei täydy pelkästään miettiä asioita jotka tapahtuvat omalla ruudulla, vaan myös mitä tietoa lähetetään verkon yli toisille pelaajille ja kuinka usein tietoa lähetetään. Peliobjektit on myös määrättävä tiettyjen pelaajien hallittaviksi luonnin aikana. Tämä on pakollista siksi, että pelilogiikka tietää minkä pelaajan objekti on kyseessä. Scripteissä on myös kriittistä määritellä, että peliobjekteja liikuttelevat vain auktoriteetin omaava pelaaja. Törmätään isoihin ongelmiin, jos kaksi eri pelaajaa liikuttelevat samaa objektia verkon yli.

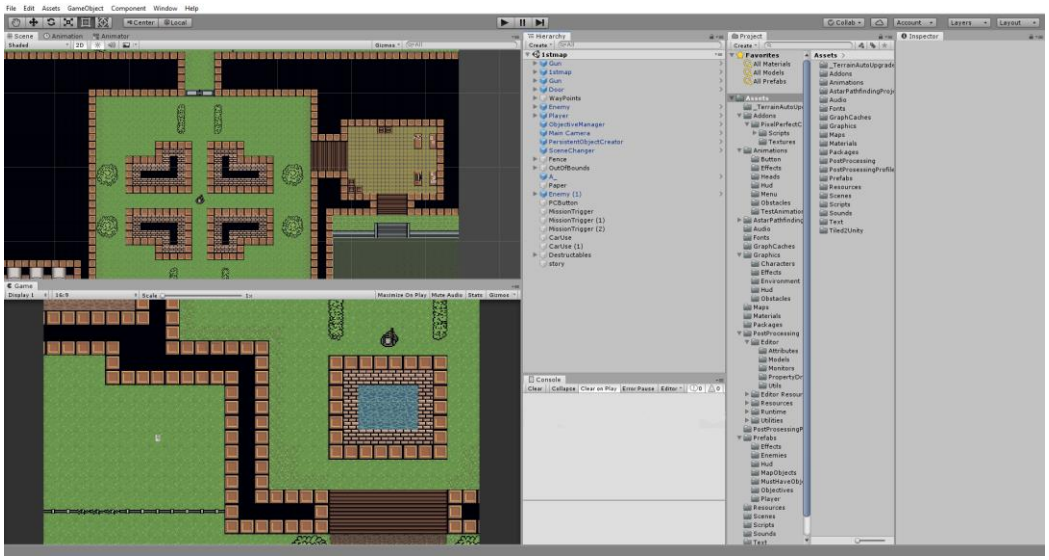
Reaaliaikaisten moninpelien teko on erityisesti hankalaa, koska peliobjektit tulee olla synkronoituna samanaikaisesti kaikilla käyttäjillä riippumatta internetin tai lähiverkon viiveestä, eli latenssista. Latenssi on aika, joka paketilta kuluu verkossa matkaan sen lähettäjältä vastaanottajalle ja takaisin lähettäjälle (TechnicalTerms 2017). Viiveestä johtuen yleensä peleissä käytetään interpolaatiota, jolla yksinkertaistettuna yritetään ennustaa peliobjektin tulevaa paikkaa ja tasoittaa sen liikettä. Pelaaja ei varsinaisesti huomaa interpolaatiosysteemiä pelatessaan jos se on tehty hyvin, mutta pelaaja taatusti havaitsee huonosti toimivan ennakoinnin. Huonon interpolaatiosysteemin tunnusmerkkejä on peliobjektien tökkiminen tai jopa kuminauhavaikutus, jossa peliobjektin paikka vaihtelee nopeasti paikasta toiseen ilman järkevää selitystä.

2 UNITY

2.1 Unity lyhyesti

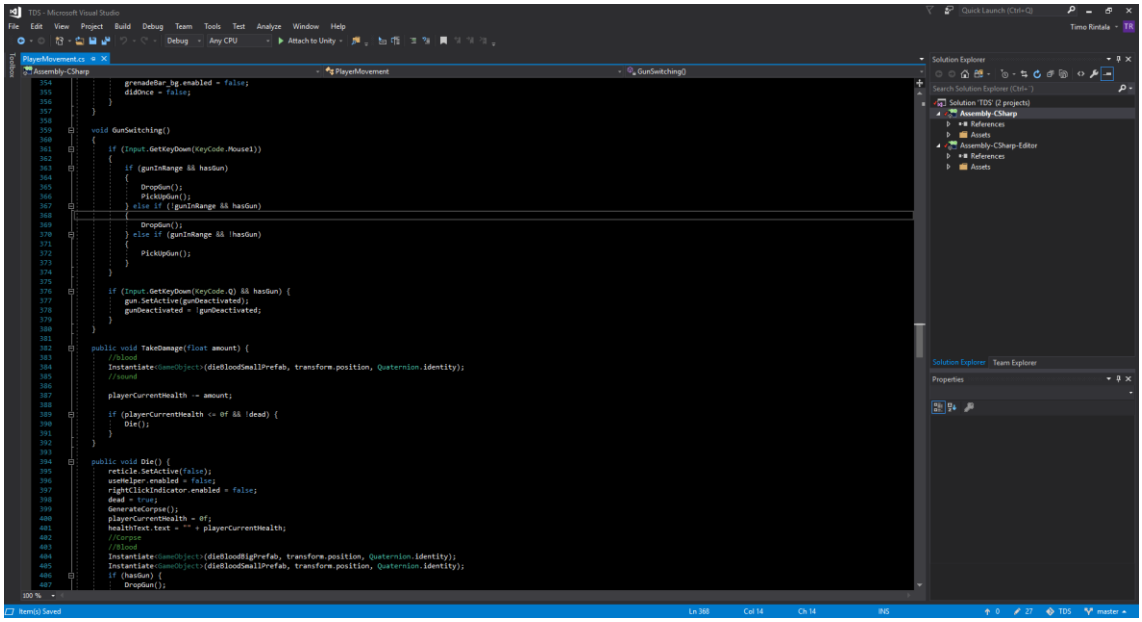
Unity on Unity Technologiesin vuonna 2005 julkaisema pelimoottori. Unityn päätarkoitus kohdistuu pelien tekemiseen, mutta Unity ei rajaa itseään käytännössä pelkästään peleihin. Unityn helposti käytettävät 3D- ja 2D-työkalut mahdollistavat ohjelmiston käytön myös pelialan ulkopuolelta. (Unity3D a 2019) Unityä voidaan käyttää esimerkiksi reaaliaikaisten elokuvien tekemiseen, arkkitehtuuriin tai rakennus- ja tekniikan alan töihin. VR (Virtual Reality) eli virtuaalitodellisuus on myös täysin tuettu Unityssä joka mahdollistaa erilaisten simulaatioiden ja pelien luomisen VR:lle. (Unity3D b 2019)

Unity julkaistiin alunperin olemaan ekslusiivisesti vain Mac OS X:n kanssa käytettävä pelimoottori (Wikipedia a 2019). Ajan mittaan kuitenkin Unity on laajentunut massiivisesti tukemaan niin montaa alustaa kuin vain on mahdollista. Vuonna 2018 tuettujen alustojen määrä on 24, joista yleisimmin käytetyt ovat Windows, Mac ja Linux sekä mobiilipuolella iOS ja Android. (Unity3D c 2019) Unityn ensimmäinen ilmaisversio julkaistiin vuonna 2009 Unity 2.6 päivityksen mukana (Unity3D d 2009). Tämä avasi pelintekemisen ovet paljon suuremmalle yleisölle, joka johti kokonaisuutena siihen, että Unitystä tuli yksi maailman suosituimmista pelimoottoreista hyvin nopeasti.



Kuva 1: Unityn käyttöliittymä

Unity koostuu graafisesta käyttöliittymästä sekä haluamasta ohjelmointiympäristöstä (IDE, Integrated development environment). Unity tarjoaa kahta erilaista ohjelmointiympäristöä, Visual Studiota ja MonoDevelopia. Molemmat ohjelmat ovat melko samanlaisia, mutta Visual Studio tarjoaa koodiin muun muassa automaattisen täydennyksen ja aaltosulkeiden parituksen, jotka tekevät käytöstä helpompaa. MonoDevelopin plussapuolena on se, että se toimii myös Linux-käyttöjärjestelmällä, kun taas Visual Studio tukee vain Windowsia ja Mac:ia.



Kuva 2: Visual Studion ohjelmointiympäristö

Unityssä koodia eli scriptejä voidaan kirjoittaa kahdella eri kielellä; JavaScriptillä ja C Sharpilla (C#). Koodikieli Boo oli myös vaihtoehtona käyttöön Unityssä aikaisemmin, mutta se poistettiin Unityn versiossa 5.0 pienten käyttäjämäärien vuoksi. Samaan aikaan C# oli selvästi yleisin käytetty koodauskieli Unityssä ja sitä käyttämällä tehtiin noin 80% kaikesta koodista Unityn sisällä. (Unity3D e 2014)

Unityn ehdoton vahvuus on sen graafinen käyttöliittymä. Ohjelmiston käyttö on pyritty tekemään mahdollisimman helpoksi käyttäjille. Unityssä kaikki tehtävät pelit koostuvat peliobjekteista ja komponenteista. Peliobjekti on yksinkertaisimmillaan vain tyhjä näkymätön piste pelimaailmassa, jolla on kaikille peliobjekteille pakollinen Transform-komponentti. Transform-komponentilla voidaan hallita peliobjektin sijaintia, rotaatiota ja kokoa. Yksinkertaisesti eri komponentteja lisää-

mällä saadaan käyttäjän haluamat ominaisuudet peliobjektiin, kuten vaikkapa objektin piirtäminen tai äänen toistaminen pelissä. Scriptit ovat myös komponentteja jotka voidaan liittää peliobjekteihin, jotta ne saadaan toimimaan. On myös mahdollista tehdä scriptejä, joita ei lisätä peliobjekteihin ollenkaan, kuten Unityn käyttöliittymää muokkaavat scriptit. Käyttäjä voi käyttää omia assettejaan pelissä. Assetti on esitysmuoto esineestä tai asiasta, jota voidaan käyttää pelissä. Erilaisia assetteja ovat esimerkiksi 3D-mallit, tekstuurit ja äänet.

Unitystä on tarjolla ilmaisversion lisäksi myös kolme erilaista maksullista versiota. Unityn maksulliset versiot ovat Unity Plus, Unity Pro ja Unity Enterprise. Verkkopelaamisen kannalta tämä on tärkeä tietää, koska ilmaisessa versiossa voi pelilläsi olla ilmaiseksi vain 20 samanaikaista käyttäjää. Unity Plus tarjoaa 50 ilmaiskäyttäjän rajan, kun taas Unity Pro:lla käyttäjiä voi olla 200. (Unity3D f 2019) Unity Enterprise on tarkoitettu isommille yrityksille ja sen ilmaiskäyttäjärajat ovat muokattu yrityksen tarpeisiin.

2.2 Unet

Unet on Unityn sisäänrakennettu verkkotyökalu, jolla on mahdollista tehdä kaiken tyylisiä verkkopelejä. Yleiset käytetyt verkkoon liittyvät lähetysprotokollat ovat UDP (User Datagram Protocol) ja TCP (Transmission Control Protocol). TCP on niin sanottu luotettava paketinsiirto, jossa varmistetaan että paketit saapuvat perille ja ovat oikeassa järjestyksessä. UDP:ssä paketin perilletuloa ei varmisteta, eli se on epäluotettava siirtotapa, joka tarkoittaa, että paketti voi kadota lähetyksen jälkeen, jos esimerkiksi paketin saajaa ei löydy. UDP ei tarvitse laitteiden välille yhteyttä, eli se on niin sanottu yhteydetön protokolla. Unet käyttää UDP-protokollaa yksinkertaisesti siksi, että se on paljon nopeampi kuin TCP. (Netscout 2014)

Yleisesti ottaen peleissä käytetään usein UDP-teknologiaa, koska sen nopeat epäluotettavat tiedonsiirrot vain parantavat pelaajakokemusta (Rouse 2018). Esimerkkinä voidaan kuvailla pelaajahahmon liikettä verkon välityksellä. Peliä ei juurikaan haittaa, vaikka yksittäinen pelaajahahmon liikuttelupaketti häviää verkossa, koska seuraava paketti korjaa kuitenkin hahmon oikeaan paikkaan.

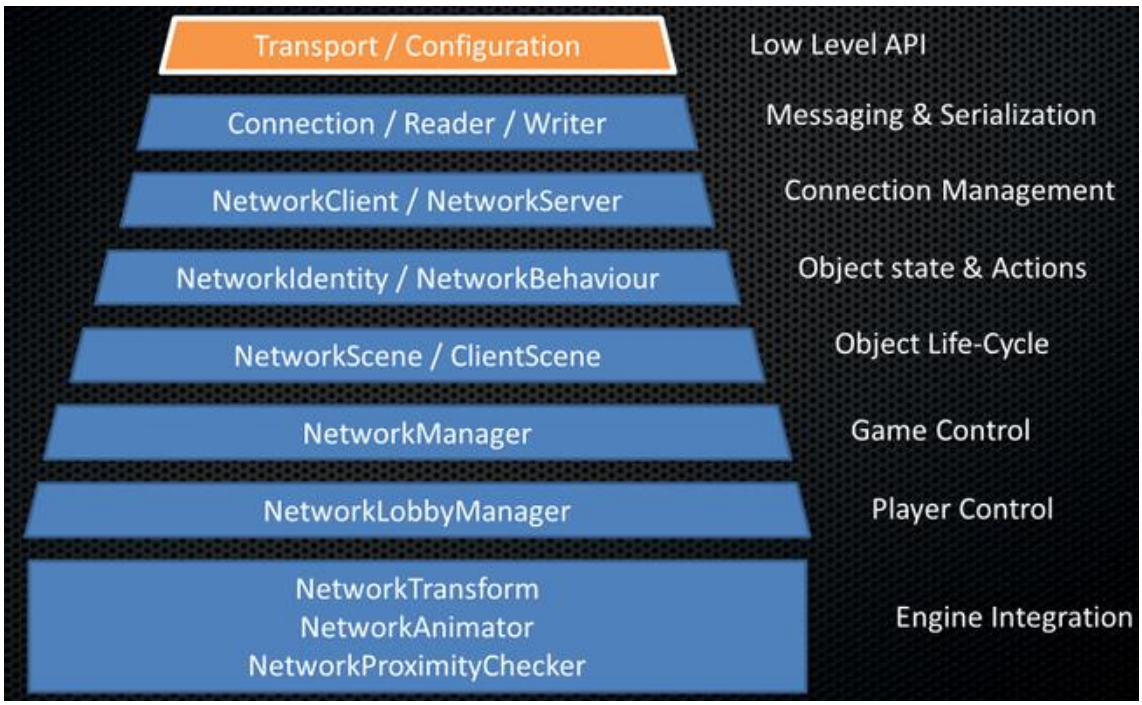
TCP:llä hävikkiä ei tapahtuisi, koska käyttäjien välille on muodostettu pysyvä yhteys, joskin protokollan hitaus vaikuttaisi pelikokemukseen massiivisesti.

Yhteenvetona voidaan todeta että TCP soveltuu tilanteisiin, joissa on tärkeä olla kaikki tiedot saatavilla ja vastaanotettuna oikeassa järjestyksessä, esimerkiksi verkkosivuilla. UDP soveltuu nopeampiin asioihin, joissa seuraava paketti korjaa edellisen paketin mahdollisen puuttumisen, esimerkiksi verkkopelit tai reaaliaikaisen videon lähettäminen. (Wikipedia b 2019)

Taulukko 1: TCP:n ja UDP:n erot (Netscout 2014)

TCP	**UDP**
Luotettava	Epäluotettava
Vaatii yhteyden	Yhteydetön
Pakettien uudelleenlähetys	Ei uudelleenlähetystä
Pakettien järjestely	Ei pakettien järjestelyä
Pakettien kuittaus	Ei kuittausta

Unityn verkkorajapintaan on rakennettu kaksi eri verkottamisen tasoa. Nämä tasot ovat matala taso (LLAPI, Low Level Application Programming Interface) ja korkea taso (HLAPI, High Level Application Programming Interface). Matala taso on käytännössä suoraan UDP:tä optimoidusti käyttävä järjestelmä, johon ei ole tehty mitään avustavia komponentteja tai scriptejä. Tämä tarkoittaa sitä, että kaikki verkkoon liittyvä täytyy koodata itse, mikä tuo peliin paljon enemmän töitä. Korkea taso taas on matalan tason päälle rakennettu valmis kehys, joka tekee verkkopelin rakentamisen helpommaksi useimmissa tilanteissa. (Unity3D g 2019)

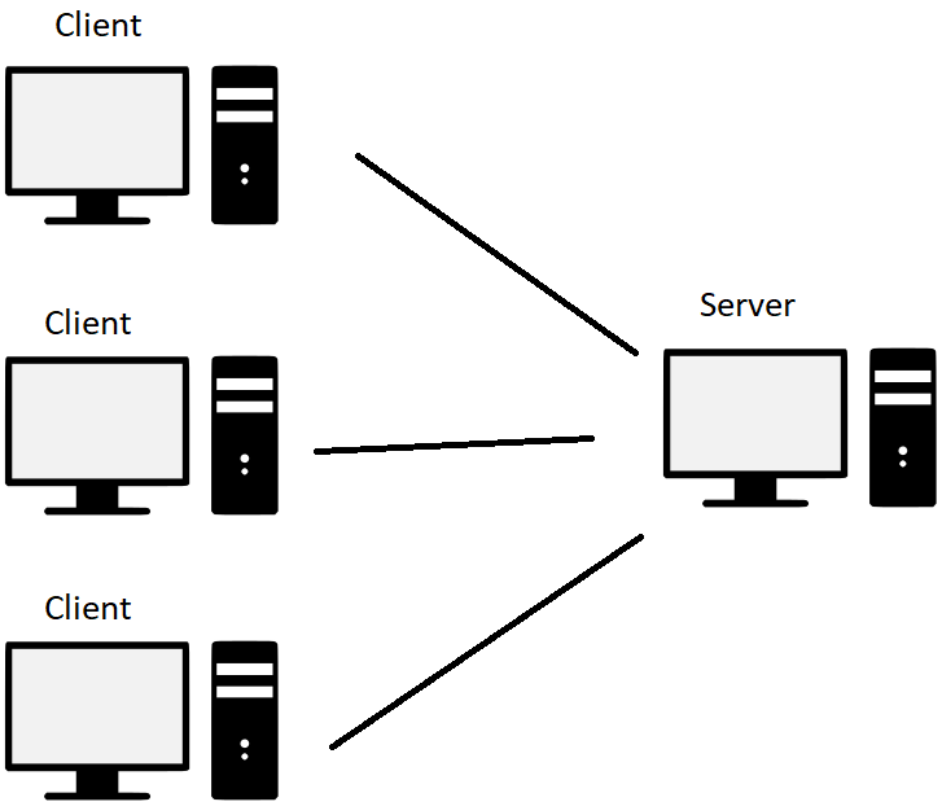


Kuva 3: Unityn verkkorajapinnan tasot (Unity3D h 2019)

Ylläolevasta kuvasta nähdään paremmin matalan tason ja korkean tason toimintaa. Oranssilla pohjalla on matala taso, kun taas kaikki korkean tason lisäämät ominaisuudet ja komponentit ovat sinisellä pohjalla. Tasoja voidaan myös käyttää sekaisin keskenään. Esimerkiksi jos yllä nähtävä NetworkTransform-komponentti ei toimi hyvin tietyssä pelissä, ohjelmoija voi rakentaa oman NetworkTransform-komponentin, johon on tehty kaikki tarvittavat asiat pelin toimimisen kannalta.

Matalaa tasoa on parasta käyttää sellaisessa tilanteessa, jossa tehdään isompia verkkoinfrastruktuureja tai edistyneempiä verkkopelejä. Korkea taso taas soveltuu parhaiten pienempiin ja yksinkertaisempiin verkkopeleihin, joihin massiivista verkkoinfrastruktuuria ei välttämättä tarvita. (Unity3D g 2019) Tässä opinnäytetyössä keskitytään korkeaan tasoon, koska se on helpompi oppia ja havainnollistaa, sekä sen pohjalta on helppo kasata ymmärrystä verkon toiminnasta yleisesti.

Unity Unetin korkea taso toimii käyttäen käyttäjä-serverimallia. Tämä tarkoittaa sitä, että käyttäjät eivät ota yhteyttä toisiinsa, vaan sen sijaan kaikki ovat yhteydessä serveriin. Käyttäjät lähettävät serverille tietoa, jonka serveri sitten lähettää edelleen takaisin kaikille käyttäjille. (Unity3D i 2019)



Kuva 4: Client-server malli (Wikipedia c 2019)

Unetin korkealla tasolla peliä tehdessä on pakko olla serveri, jolle käyttäjät voivat liittyä. Tämä serveri voi olla yksittäinen laite, jolla ei samaan aikaan pelata (Dedicated server). Myös yksi pelattava laite voi olla samaan aikaan käyttäjä ja serveri, jolle muut pelaajat sitten liittyvät pelissä. (Unity3D i 2019)

Unity Unet sisältää yksinkertaisen korkean tason graafisen käyttöliittymän, jolla voi luoda oman serverin peliin tai vaihtoehtoisesti liittyä toisen pelaajan luomaan peliin. Tätä valmiiksi luotua järjestelmää ei tulisi koskaan käyttää valmiissa pelissä, vaan se on tarkoitettu lähinnä helpottamaan ja nopeuttamaan prototyyppivaihetta. Tämä komponentti esitellään myöhemmin tarkemmin.

Unity Unet sisältää myös Unityn tarjoamat matchmaker-palvelut sekä internetissä toimivat serverit. Matchmakerin tarkoitus on sovittaa samantasoiset pelaajat pelaamaan keskenään, jos pelistä halutaan kilpailullisempi. Matchmakeria voidaan

myös yksinkertaisemmin käyttää siihen, että pelaajat laitetaan liittymään automaattisesti peliserverille, eikä pelaajan itse tarvitse valita haluamaansa peliä listalta. (Unity3D g 2019)

Unity Unetin ilmaisversiossa yhdellä pelillä voi olla 20 samanaikaista käyttäjää ilman kustannuksia (Unity3D f 2019). Tämä luku riittää hyvin prototyyppaamisvaiheeseen sekä pienimuotoiseen testaamiseen, eikä maksuja tarvita. Isompaan testaamiseen saattaa olla hyödyllistä maksaa lisäkäyttäjistä, mutta Unity Unet on suhteellisen kallis vaihtoehto tässä asiassa verrattuna muihin vaihtoehtoihin. Tästä johtuen Unity Unet soveltuu mielestäni parhaiten nimenomaan opettamaan verkottamista ja nopeaan prototyyppaamiseen.

3 Verkkotyökalujen vertailu

3.1 Kilpailijat

Tässä luvussa käydään läpi Unity Unetin kilpailijoita ja niiden ominaisuuksia. Unityn kanssa käytettäviä verkkoratkaisuja on saatavilla toista kymmentä ja niiden kaikkien lävitse kahlaaminen tässä opinnäytetyössä olisi mahdotonta. Vertailuun valitsin tunnetuimmat ja yleisimmät Unityn kanssa käytetyt verkkoratkaisut. Nämä ovat Photon Unity Networking, Photon Bolt, Forge Networking ja Darkrift 2 Networking. Useimmiten vähemmän tunnetuissa ja käytetyissä verkkoratkaisuissa on se ongelma, että dokumentaatio on puutteellinen tai vaikeasti ymmärrettävä. Lisäksi vähemmän tunnetuista ratkaisuista on vaikeampi löytää koodiesimerkkejä ja ylipäätään oppimisaika pitenee huomattavasti, koska kaikki pitää kokeilla ja löytää itse.

3.1.1 Photon Unity Networking

Photon Unity Networking (PUN) on Exit Gamesin luoma verkkoratkaisu Unitylle. Se on yksi Photon-tuoteperheen tunnetuimmista ohjelmistoista. PUN on käytännössä klooni alkuperäisestä Unityn verkkotamisen ohjelmointirajapinnasta, vahvistettuna Photonin tuomalla infrastruktuurilla (Photon a 2019). PUN:in toiminta perustuu Photonin reaaliaikaisen pilvipalvelun sekä P2P-systeemin (Peer-To-Peer) yhteistyöhön (Photon b 2019). Peer-To-Peer tarkoittaa käyttäjältä käyttäjälle toimivaa pakettiensiirtoa, jossa varsinaista serveriä ei tarvita ja se ei vastaanota tai lähetä mitään tietoa.

Peliserveri luodaan Photonin pilvipalveluun, mutta pelilogiikka pyörii täysin pelaajien välillä, joista yksi on nimellinen serveri muille käyttäjille. PUN:ssa ei tarvitse ostaa omia servereitä, koska kaikki logiikka toimii pilvessä, joka tarkoittaa sitä, että ylläpidon hinta pienenee huomattavasti. (Photon b 2019)

PUN on arkkitehtuuriltaan erittäin yksinkertainen ja helposti ymmärrettävä kokonaisuus. PUN käyttää RPC-systeemiä (Remote Procedure Call) datan lähetyksessä muille pelaajille (Photon a 2019). Yksinkertaisuudessaan RPC on tapa kommunikoida muille käyttäjille ajamaan jokin tietty metodi omasta koodista.

PUN sopii täydellisesti yksinkertaisemmille peleille, kuten esimerkiksi verkkosivuilla pyöriville pienille verkkopeleille tai peleille, jotka eivät tarvitse isoa serverilogiikkaa toimiakseen (Youtube 2017). PUN:in yksi parhaista ominaisuuksista on se, että se tukee täysin pelin luojan vaihtumista. Tämä tarkoittaa sitä, että jos kyseisen pelin luoja poistuu pelistä kesken kaiken, serveri ei kaadu vaan valitsee pelaajien joukosta uuden master-käyttäjän, joka on nimellinen serverin luoja (Photon a 2019).

PUN:ista on tarjolla täysin ilmainen sekä maksullinen versio. Ilmaisessa versiossa on mahdollista olla 20 samanaikaista käyttäjää, kun taas maksullisessa PUN+:ssa saa 60 kuukaudeksi 100:n samanaikaisen käyttäjän rajan. Lisäksi on mahdollista ostaa myös pelkkää korotusta samanaikaisten käyttäjien määrään. (Photon a 2019) PUN:in dokumentaatio on myös erinomaisesti järjestelty ja kirjotettu, sekä sen ymmärtäminen on helppoa. PUN:issa on myös tarjolla paljon valmiita esimerkkiprojekteja, jolla oppimista voidaan tehostaa entisestään helposti.

3.1.2 Photon Bolt

Photon Bolt on myöskin Exit Gamesin luoma verkkoratkaisu Unitylle ja se on yksi Photon-tuoteperheen tunnetuimmista ohjelmistoista Photon Unity Networkingin lisäksi. Yleensä pelin tyylistä ja infrastruktuurista riippuen valitaan joko Photon Unity Networking tai Photon Bolt, riippuen siitä kumpi sopii paremmin kyseiseen peliin. Photon Bolt toimii serveri-käyttäjä järjestelmällä, jolloin pelillä on pakollista olla serveri jolle liittyä (Photon a 2019).

Serveri voi olla käyttäjä samanaikaisesti, mutta jos kyseinen serverikäyttäjä poistuu pelistä, myös serveri kaatuu samalla. Minkäänlaista pelin luojan vaihtumista ei tueta. Tästä syystä tämän kaltaisissa peleissä yleensä käytetään dedicated servereitä, jotka ovat palvelimia erillisillä laitteilla, joissa ei ole käyttäjää samalla

pelaamassa. Dedicated serverit takaavat serverin pystyssäpysymisen kellon ympäri ja käyttäjät voivat liittyä ja poistua halutessaan. (Photon a 2019)

Photon Bolt on korkeamman tason ohjelmointirajapinta, joka antaa määritellä verkottamista erilaisten datastruktuurien kautta. Photon Bolt ei perustu Photon Unity Networkingissa käytettyyn RPC-systeemiin, vaan hoitaa samat asiat event-systeemillä. Event-systeemi eroaa RPC-systeemistä siten, että eventit eivät ole millään tavalla sidonnaisia yksittäisiin peliobjekteihin. (Photon c 2019) Tämä tarkoittaa sitä, että voidaan rakentaa erillinen objekti joka hoitaa isommat verkkoon liittyvät kokonaisuudet samasta paikasta, eikä jokainen peliobjekti hoida omaa osaansa, eli yhdestä objektista voidaan kontrolloida useiden peliobjektien asioita ja isompia kokonaisuuksia.

Photon Bolt on tarkoitettu skaalaltaan isommille peleille, jotka vaativat hyvin toimiakseen kunnollisen serverilogiikan. Tämän tapaisia pelejä ovat muun muassa ensimmäisen tai kolmannen persoonan ammuntapelit. Serverilogiikka on tärkeä huijausohjelmien estämisen takia, koska silloin kaikki koodi ei toimi vain paikallisella laitteella, vaan osa siitä ajetaan serverin kautta. (Photon c 2019) Lisäksi Photon Boltissa on valmiiksi ominaisuuksia, jotka auttavat esimerkiksi viiveen kompensoinnissa ja objektien liikkeiden ennakoinnissa (Photon a 2019). Edellä mainitut ominaisuudet ovat hyvin tärkeitä erityisesti nopeiden toimintapelien toiminnallisuuden kannalta, jotta käyttäjällä on paras mahdollinen pelikokemus.

Photon Boltista on myös tarjolla täysin ilmainen versio 20 samanaikaisen käyttäjän rajalla ja maksullisen version rajat ovat samat, kuin PUN:ssa (Photon a 2019). Photon Boltin dokumentaatio on myös erinomainen kuten PUN:inkin, joten siitä on paljon apua varsinkin aloittelijoille.

3.1.3 Forge Networking Remastered

Forge Networking Remastered on Bearded Man Studiosin tekemä verkkoratkaisu Unitylle. Alkuperäinen Forge Networking julkaistiin kesäkuussa 2015 ja se oli maksullinen kaikille käyttäjille (Forge a 2015). Kaksi vuotta myöhemmin vuoden 2017 kesäkuussa julkaistiin Forge Networking Remastered ja se siirtyi samalla

vapaaseen lähdekoodiin ja näin ollen siitä tuli ilmainen kaikille halukkaille (Forge b 2017). Vapaa lähdekoodi helpottaa kehittäjiä esimerkiksi bugien korjaamisessa, koska käyttäjät voivat itse tehdä bugikorjauksia.

Forge sallii rajoittamattomasti samanaikaisia käyttäjiä, mutta sillä ei ole omia servereitä. Pelintekijän pitää siis itse luoda serveriteknologia tukemaan peliä. (StackExchange 2017) Forgen käytön alussa ei tarvitse miettiä sopiiko kyseisen pelin tekeminen juuri Forgelle, koska se tukee serveri-käyttäjä mallia sekä myös pelaajalta pelaajalle siirtyvää pakettiensiirtoa. Forgen toimintaperiaate on hyvin samanlainen kuin Unity Unetin ja Photon Unity Networkingin, sillä se käyttää myös Remote Procedure Calleja (RPC) pakettien siirrossa käyttäjiltä toisille (Forge c 2017). Forgesta löytyy myös jonkin verran valmiita verkottamisen osia nopeaa prototyyppaamista varten, mutta valikoima on huomattavasti pienempi, kuin Unity Unetissa.

Vaikka Forge Networking Remastered onkin ilmainen, se ei sovi hyvin opiskelijoiden nopeille projekteille. Isoin syy tähän on se, että serverien asentamiseen ja säätämiseen ei ole ylimääräistä aikaa hukattavaksi. Lisäksi Forgen dokumentaatio kärsii vapaasta lähdekoodista eikä välttämättä kaikkiin ongelmiin löydy vastausta. Forgella kuitenkin on hyvin toimiva asiakastuki, josta ongelmiin saa ratkaisun pitkällä tähtäimellä, mutta lyhyellä aikavälillä se on huono ongelmien tapahtuessa.

3.1.4 Darkrift 2 Networking

Darkrift 2 on Jamsterin julkaisema verkkotyökalu Unitylle. Alkuperäinen Darkrift julkaistiin huhtikuussa 2015 (Darkrift a 2015). Myöhemmin kuitenkin uudistettu Darkrift 2 syrjäytti sen maaliskuussa 2018 (Darkrift b 2018). Darkrift 2:n valttikortti on sen nopeus ja tehokkuus. Se on yksi parhaiten optimisoituja verkkotyökaluja markkinoilla tällä hetkellä ja sen pyörittäminen tietokoneella vie erittäin vähän resursseja (Darkrift c 2018).

Valitettavasti Darkrift 2 ei sisällä minkäänlaisia valmiita työkaluja, joilla prototyypata. Tästä johtuen se ei ole varsinaisesti hirveän aloittelijaystävällinen valinta,

koska kaikki asiat pitää ymmärtää ja tehdä itse alusta alkaen. Lisäksi Darkrift 2 on vain yhden kehittäjän luoma, joten esimerkiksi työkalun bugien korjaaminen tai uusien ominaisuuksien kehittäminen saattaa kestää erittäin pitkään (Unity3D j 2015).

Dokumentaatio on yllättävän hyvin tehty vaikka tekijöitä onkin vain yksi, mutta dokumentaatiostakin puuttuu uusimmat verkkotyökalun päivitykset lähes kokonaan. Darkrift 2:sta on tarjolla myös maksullinen Pro-versio joka lisää muutamia ominaisuuksia, mutta ilmaisversiossa on kaikki olennaiset ominaisuudet verkkopelien tekemiseen jo valmiina. Maksullisen version lisiä ovat muun muassa matchmaking, laajennetut käyttäjien statistiikat ja keskusteluihin sisältyvä kiroilunesto. (Darkrift c 2018)

Darkrift 2 on erittäin monipuolinen ja tehokas työkalu, mutta se vaatii tietoa verkottamisesta sekä Microsoftin .NET frameworkista. Samanaikaisia käyttäjiä Darkrift 2:ssa voi olla rajaton määrä, mutta serverien pystyttäminen ja ylläpito jää tässäkin tapauksessa kehittäjän vastuulle. (Darkrift c 2018) Näin ollen Darkrift 2 ei sovellu vasta-aloittelijalle, vaan markkinoilla on parempia vaihtoehtoja ensimmäisen projektien tekoon tai prototyyppaamiseen kuten Unity Unet tai Photon Unity Networking.

3.2 Yhteenveto kilpailijoista

Kaikki viisi ylhäällä esiteltyä verkkotyökalua (Unity Unet, Photon Unity Networking, Photon Bolt, Forge Networking Remastered, Darkrift 2) ovat erinomaisia, mutta ne on tarkoitettu hieman erilaisiin tilanteisiin riippuen tehtävän pelin koosta, budjetista ja tyypistä. Forge ja Darkrift 2 on tarkoitettu isompiin projekteihin, joissa aikaa ja budjettia on enemmän ja joissa verkon optimointi on suuremmassa roolissa. Valitettavasti Forgesta ja Darkrift 2:sta puuttuu valmiit serverit, mikä käytännössä eliminoi näiden työkalujen käytön opiskelijoiden nopeissa projekteissa, koska aikaa on yleensä erittäin vähän ylimääräiseen työskentelyyn.

Tämä jättää jäljelle Unity Unetin, Photon Unity Networkingin ja Photon Boltin. Kaikilla kolmella tuotteella on omat sisäänrakennetut serverit ja niillä on suhteellisen

helppo aloittaa työskentely, mutta Unity Unetin korkean tason työkalut ovat kaikkein helppokäyttöisimmät aloittelijalle. Valmiit työkalut tuovat oppimiseen ja prototyyppaamiseen nopeutta, koska kaikkea ei tarvitse tehdä alusta alken itse. Lisäksi myös kaikista Unity Unetin korkean tason ominaisuuksista on tarjolla ilmaiseksi lähdekoodi, josta voi tutkia miten ne on tehty ja oppia lisää. Photonin tuotteiden edut ovat erittäin laaja valikoima hyvälaatuisia servereitä sekä melko halvat hinnat jos peliä haluaa lähteä skaalaamaan isommaksi ajan myötä. Photonin tuotteet ovat myös hyvin suosittuja, jolloin luonnollisesti ongelmatilanteisiin löytyy foorumeilta enemmän tietoa.

Oma suositukseni aloittelijalle menee Unity Unetille. Sen ominaisuudet tukevat kaikista parhaiten nopeaa prototyyppaamista ja oppimista. Yleiset konseptit verkottamisessa ovat pitkälti samoja, joten perusteiden opettelun jälkeen työskentely sujuu pienellä vaivalla millä tahansa yllä esitellyllä työkalulla. Photonin tuotteet ovat kuitenkin todella hyvin tehty ja oma mielipiteeni on se, että verkottamisen perusteiden opettelun jälkeen Unetin kautta kannattaa siirtyä suoraan jompaankumpaan Photonin tuotteista. Tietysti vaihtoehtona on myös Forge tai Darkrift 2 jos peliin tarvitaan hyvin optimoitu verkkoliikenne tai budjetti on isompi.

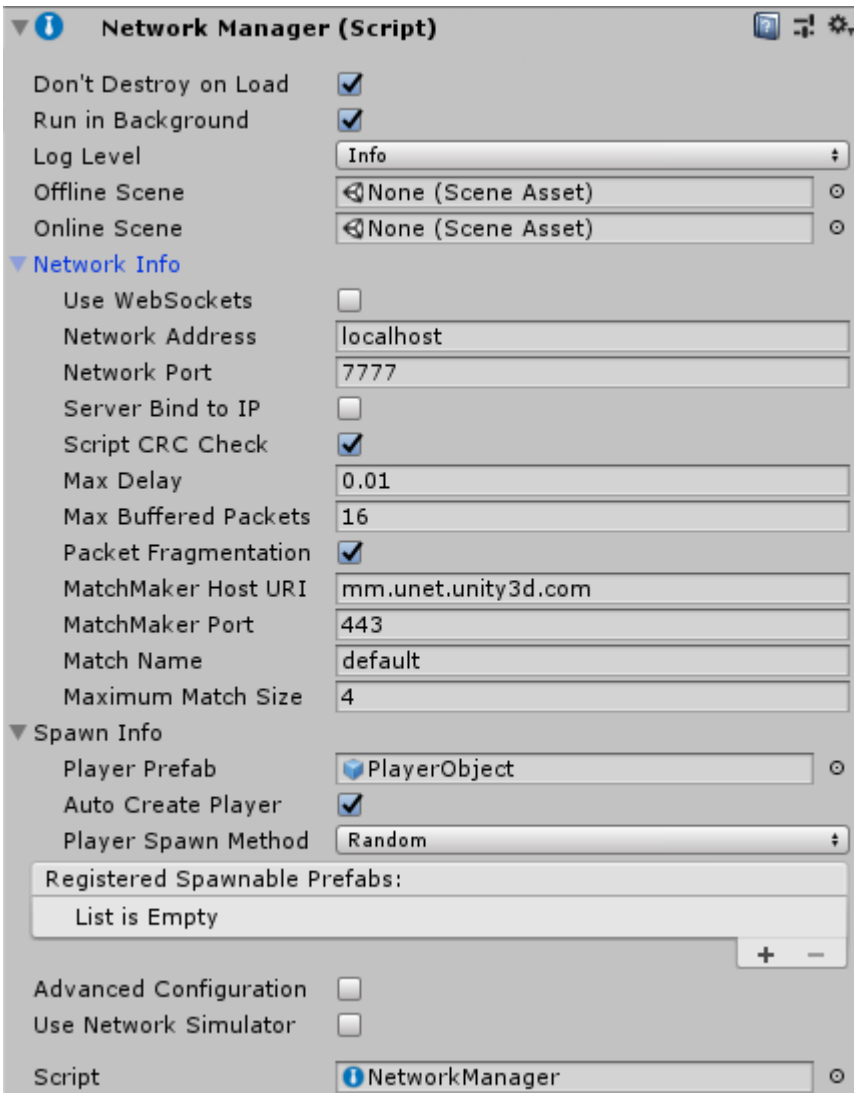
4 UNITY UNETIN KÄYTTÄMINEN

4.1 Tärkeät komponentit

Tässä kappaleessa käydään läpi kolme tärkeintä komponenttia Unity Unetin korkeaa tasoa käytettäessä; NetworkManager, NetworkManagerHUD sekä NetworkIdentity. NetworkManager sekä NetworkIdentity on lähes tulkoon pakollisia käyttää, koska ilman niitä tärkeät toiminnallisuudet pitäisi tehdä alusta alken itse, joka vie näiden kahden tapauksessa erittäin paljon aikaa. NetworkManagerHUD:in käyttö ei ole pakollista, vaan sen tilalle voidaan helposti tehdä toiminnallisuudet omin käsin. Prototyyppaamiseen kuitenkin NetworkManagerHUD on mainio työkalu ja se tuo paljon nopeutta tekemiseen. Kannattaa kuitenkin muistaa että NetworkManagerHUD:ista on tehty tarkoituksellisesti suhteellisen ruma, jotta käyttäjä tekisi oman käyttöliittymän sen päälle eikä päätyisi käyttämään valmiissa pelissä tätä ratkaisua.

4.1.1 NetworkManager

NetworkManager on Unity Unetin korkean tason sydän. Se luo pohjan koko systeemille, jolla kontrolloidaan muita korkean tason komponentteja (Unity3D k 2019). NetworkManageria varten kannattaa yleensä luoda oma tyhjä objekti Unityssä, johon lisätään NetworkManager -komponentti. Varsinaisesti mikään ei estä komponentin lisäämistä johonkin muuhun objektiin, mutta pitää muistaa, että jos NetworkManagerin sisältämä objekti tuhotaan, katkeaa myös verkkopeli samanaikaisesti. NetworkManagerin objekti kannattaa luoda jo ennen peliä jonkinlaisessa aloitusscenessä tai vaihtoehtoisesti valikossa moninpeli -nappia painettaessa. NetworkManager ei kuluta tietokoneelta juuri mitään silloin kun sitä ei käytetä, joten ei haittaa vaikka se onkin taustalla yksinpelinkin aikana.



Kuva 5: NetworkManager komponentti

NetworkManager komponentissa on paljon eri kohtia, joita voi muuttaa sen sisältä. Kuitenkin aloitusasetukset on hyvin säädetty jo valmiiksi komponentissa, jolloin useimmat kohdat voi jättää niille asetuksille, jotka niihin on jo valmiiksi asetettu. Ensimmäiset kaksi rastia on tärkeitä olla päällä. Don't Destroy on Load -kohta tarkoittaa sitä, että objektia ei tuhota, vaikka Unityn sisäinen scene vaihtuisikin toiseen. Näin ollen voidaan pelissä esimerkiksi vaihtaa karttaa ilman että verkkoyhteys katkeaa.

Run in Background on myös hyvin tärkeä, koska ilman sitä jos pelin ikkuna ei ole kohdennettuna, verkkoyhteys katkeaa. On siis tärkeää pitää verkkoyhteys päällä koko ajan vaikka peli ei olisi käyttöjärjestelmässä päälimmäisenä kohdistettuna.

Log levelistä voi säätää lokitietojen tyyppiä mikä ei ole kovinkaan tärkeä ominaisuus. Offline Scene ja Online Scene on tarkoitettu peleille, joissa yksinpeli ja moninpeli on erotettuna pelissä ja niille on omat valikkonsa.

Network Info kohdan alta löytyy kaikki itse serveriin ja sen ominaisuuksiin liittyvät tiedot. Websocketteja käytetään lähinnä selainpeleissä, joten niitä ei käytetä tämän työn esimerkeissä. Network Address on serverin osoite joka luodaan tai johon yhdistetään pelaaja ja Network Port on serverin osoitteen portti. Suurin osa muista tiedoista on valmiiksi niin kuin pitää. Halutessaan kehittäjä voi tehdä oman matchmakerin, jolloin sen osoite ja portti laitetaan MatchMaker Host URI ja Port kohtaan. Serverin nimeä ja kokoa voidaan säätää kahdella viimeisellä valikolla.

Spawn Infon alla on vähän kohtia, mutta ne ovat erittäin tärkeitä, jotta vältytään virheilmoituksilta. Player Prefab kohtaan pitää lisätä prefabi, jolla on NetworkIdentity komponentti liitettynä. NetworkIdentity komponentti selvennetään myöhemmin tarkemmin. Player Prefabiin lisättävä objekti on siitä tärkeä, että se luodaan automaattisesti pelaajan tehdessä serveri tai sellaiselle liityttäessä. Kaikki verkossa liikuteltavat objektit pitää lisätä viimeiseen Registered Spawnable Prefabs kohtaan. Niissä myös täytyy olla NetworkIdentity komponentti liitettynä, muuten niitä ei voi lisätä listaan.

Player Prefabin käyttöön on kaksi erilaista käytäntöä. Ensimmäinen käytäntö on se, että pelaajan liikuteltava objekti pelissä on Player Prefabiin lisättävä objekti. Toinen käytäntö on taas se, että Player Prefabin objekti pitää sisällään kaikki tiedot pelaajasta omassa scriptissä ja kyseinen scripti luo pelaajalle objektin pelimaailmaan Registered Spawnable Prefabsin kautta. Ensimmäinen metodi tulee esiin joidenkin tutoriaalien joukossa, mutta oma mielipiteeni on se, että kyseinen metodi on huono. Player Prefab objektin ollessa myös pelaajaobjekti, mitä tapahtuu silloin, kun esimerkiksi kuolet pelissä ja peliobjektisi tuhotaan? Player Objekti pitää olla jokaisella pelissä olevalla pelaajalla ja jos se tuhotaan kesken kaiken saadaan aikaan paljon ongelmia ja verkkopelin katkeaminen. Lisäksi ensimmäisellä metodilla on ongelmana pelit, joissa pelaaja ei ole pelissä yksittäinen objekti, kuten vaikkapa strategiapeleissä, joissa samalla pelaajalla voi olla useita eri hahmoja liikuteltavana.

On siis suositeltavaa että Player Prefab objektiksi laitetaan erillinen tyhjä objekti, jonka sisäisessä scriptissä on kaikki tämän kyseisen pelaajan tarvittavat tiedot eikä sillä objektilla tehdä mitään muuta. Tällä systeemillä vältytään kaikista varmimmin ongelmatapauksilta pitkällä tähtäimellä, kun Player Prefab objektia ei vahingossakaan tuhota käytön aikana.

Kaksi viimeistä kohtaa NetworkManagerissa on myös välillä eteen tulevia, mutta ei kovinkaan tärkeitä kohtia prototyyppiä tehdessä. Advanced Configurationista saa verkkoon liittyvät lisäasetukset auki, mutta ne ovat hyvin säädetyt jo valmiiksi peruskäyttöä varten. Use Network Simulator -kohtaa voidaan käyttää simuloimaan isompia vasteaikoja pelin sisällä, joka on hyödyllinen toiminto testausta varten.

Pelin rakenteesta johtuen on aina helpompi tehdä moninpelistä yksinpelattava versio, kuin toisinpäin. Jos yksinpeliin yritetään lisätä moninpeli, joutuu käytännössä koko koodin kirjoittamaan alusta alkaen uusiksi ottamaan huomioon verkkopelaaminen. Vastaavasti moninpeliin lisättäessä yksinpelin mahdollisuus, ei tarvitse koodin puolelta kuin asettaa pelaaja omalle serverilleen yksin ja estää muiden käyttäjien pääsy peliin. Tällöin voidaan käyttää kaikkia verkkopeliin tehtyjä ominaisuuksia, vaikka pelataankin ilman viivettä omalla serverillä yksin.

4.1.2 NetworkIdentity

NetworkIdentity on toinen tärkeimmistä Unity Unetin korkean tason komponenteista. Se ei ole kovin monimutkainen käytettävä, mutta sitäkin tärkeämpi. Tätä komponenttia käytetään kaikissa verkon yli lähetettävissä objekteissa. Sen päätehtävä on pitää muistijälki objekteista, joissa se on kiinni, jotta kaikki clientit tietävät että se on sama objekti (Unity3D l 2019). NetworkIdentity komponentin omaavat objektit pitää aina olla NetworkManagerissa Player Prefab kohdassa tai Registered Spawnable Prefabs kohdassa, jotta ne pystytään luoda serverille pelin aikana. NetworkIdentity komponentin omaavat objektit pitää myös aina luoda jälkikäteen, scenessä ei voi olla valmiina näitä objekteja (Unity3D l 2019). Syy tälle on yksinkertaisesti se, että objekteja ei saada siirrettyä oikeiden pelaajien haltuun, jos ne on luotu ennen pelaajien liittymistä serverille.

Kuva 6: Network Identity komponentti

Kuten ylläolevasta kuvasta voidaan nähdä, asetuksia komponentilla ei ole montaa. Asetukset ovat toisensa poissulkevia, eli jos toinen on päällä toisen on pakko olla poissa päältä. Ensimmäinen kohta on tarkoitettu vain serverille luotaviin objekteihin. Tällöin vain serverillä on valta merkittyihin objekteihin. Local Player Authorityn merkitys taas on erittäin tärkeä tietää.

Useimmissa tapauksissa Local Player Authority halutaan kytkeä päälle. Local Player Authority on kytkettynä päälle objekteissa siksi, että sillä asetetaan eri objekteille eri käyttäjät isänniksi. Näille objekteille voi antaa käskyjä vain se käyttäjä, jolla on auktoriteetti kyseiseen objektiin. Tämä asia selviää paremmin myöhemmin, kun käydään läpi serverille lähetettäviä komentoja ja sieltä takas tulevia ClientRpc -viestejä ja niiden toimintaa.

Unityssä normaaleilla peliobjekteilla on aina tallennettuna oma muistipaikka tietokoneella, josta niiden tiedot löytyvät. Tällä muistipaikalla etsittäessä objektit löytyvät aina kunhan on kyse vain yhdestä tietokoneesta. (Unity3D m 2019) Iso ongelma tämän kanssa on se, että jos halutaan sama objekti kahdelle eri tietokoneelle samanaikaisesti verkon välityksellä, niiden muistipaikat molemmilla tietokoneilla on eri paikoissa. Tämä on syy minkä takia tarvitaan NetworkIdentityä. Se lisää jokaiseen käytettyyn objektiin oman uniikin Network ID:n, jonka kautta eri tietokoneet voivat tunnistaa sen tietyn objektin ja näin ollen etsiä muistipaikan omalta tietokoneelta (Unity3D m 2019).

4.1.3 NetworkManagerHUD

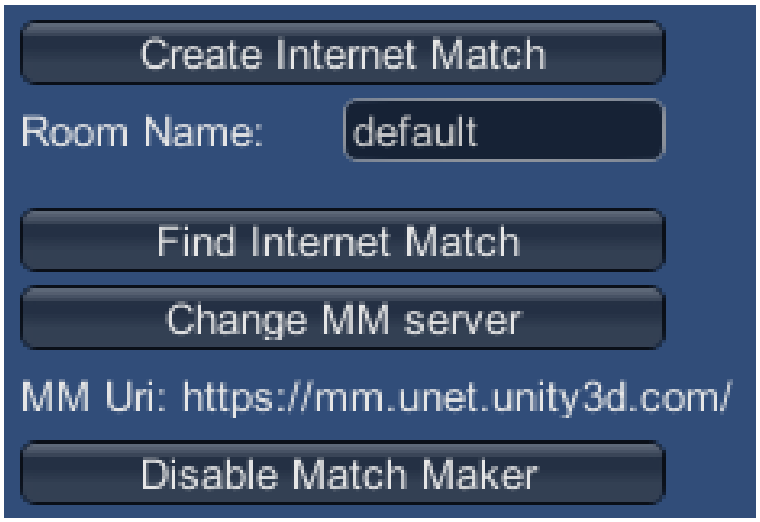
NetworkManagerHUD on nopeaan prototyyppaamiseen tarkoitettu käyttöliittymä, jonka päätehtävänä on saada pelin käyttäjät samalle serverille helposti ja nopeasti (Unity3D n 2019). Käyttöliittymän avulla voidaan luoda paikallisia servereitä

sekä myös liittyä niille. Matchmakerin avulla voidaan myös helposti tehdä internetissä toimiva serveri Unityn omille servereille sekä liittyä niille toisilla käyttäjillä.

Kuva 7: NetworkManagerHUD komponentti

Nappulat käyttöliittymässä ovat hyvin yksinkertaiset. LAN Host napilla luodaan serveri johon myös itse liitytään käyttäjänä. LAN Server Only napilla tehdään vain serveri, mutta ei liitytä siihen, vaan se toimii niin sanottuna dedicated serverinä. LAN Client nappia painettaessa liitytään paikallisverkon serverille, jonka osoite laitetaan viereiselle osoiteriville. Testatessa tämä on vain localhost.

Verkkopelien testaamista helpottaa se, että voidaan hyvin helposti testata peliä kahdella käyttäjällä samanaikaisesti vaikka käytössä olisi vain yksi tietokone. Yksi käyttäjä tehdään tekemällä buildi pelistä ja ajetaan peli tietokoneelta sen jälkeen, kun taas toinen käyttäjä pelaa suoraan Unityn kautta.



Kuva 8: Matchmaker ikkuna

Ylläolevasta kuvasta nähdään miltä näyttää ruutu, kun painaa Enable matchmaker -nappia käyttöliittymästä. Create Internet Match luo serverin, jonka huoneen nimi valitaan alapuolelta. Find Internet Match etsii nettiservereitä ja luo listan niistä, joista voi päättää mille liittyy. Change Matchmaking server vaihtoehtoa ei tarvita, jos käytetään Unityn luomia servereitä, mutta sillä voi vaihtaa matchmakerin serveriä vaihtoehtoisesti jopa omiin servereihin.

4.2 Komennot (Cmd)

Komentoja eli Commandeja (Cmd) käytetään liikuttelemaan dataa pelaajien clienteiltä serveriä päin (Unity3D o 2019). Komennot ovat erittäin tärkeitä, eikä ilman niitä pysty rakentamaan Unity Unetillä verkkopeliä. Komennot tehdään aina omaan metodiinsa ja niillä on muutama sääntö mikä pitää muistaa. Komentometodin yläpuolelle merkitään aina [Command] merkiksi siitä, että kyseinen metodi on komento. Myös saman metodin pitää alkaa kirjainyhdistelmällä Cmd, muutoin saadaan virheilmoitus Unityltä.

Komentoja voivat lähettää vain ja ainoastaan clientit. Clienteillä täytyy myös olla auktoriteetti siihen objektiin, josta komento lähetetään, muutoin se ei tee mitään. Auktoriteetti objektille annetaan NetworkIdentity komponentin kautta rastittamalla Local Player Authority.

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

public class PlayerObject : NetworkBehaviour
{

    public GameObject playerUnit;

    void Start()
    {

        //If we dont have authority over the gameobject, return out of it
        if (!hasAuthority) {
            return;
        }

        CmdSpawnMyOwnUnit();
    }

    //Server command, need to have authority
    [Command]
    void CmdSpawnMyOwnUnit() {
        GameObject g = Instantiate(playerUnit);
        NetworkServer.SpawnWithClientAuthority(g, connectionToClient);
    }
}
```

Kuva 9: Commandien käytön esimerkki

Verkko-objektien koodia kirjoittaessa pitää muistaa muutama asia ennen alkua. Tarvitaan verkottamisen kirjasto käyttöön rivillä using UnityEngine.Networking, sekä luokka pitää periä MonoBehaviourin sijaan NetworkBehaviourista. NetworkBehaviour periytyy myös MonoBehaviourista, joten kaikki NetworkBehaviourin toiminnallisuudet toimii silti aivan normaalisti.

Yllä olevassa kuvassa esitellään miten luodaan pelaajalle oma visuaalinen yksikkö serverikomennon kanssa. Koodinpätkä on peräisin NetworkManageriin Player Prefabiin tehdystä omatekemästä scriptistä. Tässä tapauksessa haluamme Player Prefabin olevan vain tyhjä objekti pelissä ja haluamme että se luo automaattisesti oman liikuteltavan pelaajaobjektimme peliin liityttäessä.

Verkko-objekteja luodessa pitää muistaa se, että usein samoja koodeja käyttäviä yksiköitä on monta samassa scenessä mutta eri käyttäjillä. Siksi on tärkeää yksilöidä se, että mikä objekti koodia saa käyttää. Kohta if (!hasAuthority) on hyvin

tärkeä ylläolevassa kuvassa, koska muutoin kaikki scenessä olevat Player Prefabit loisivat toisilleen lisää ja lisää pelaajaobjekteja. Haluamamme käytös on se, että jokaisen pelaajan clientti pyytää itse serveriä luomaan itselleen objektin.

Komentometodi on hyvin yksinkertainen sisällöltään, siinä luodaan omalle tietokoneelle objekti Instantiate -komennolla, jonka jälkeen pyydetään serveriä luomaan sama objekti muillekin pelaajille näkyväksi komennolla NetworkServer.SpawnWithClientAuthority. SpawnWithClientAuthority ottaa vastaan kaksi parametriä, ensimmäisenä peliobjektin ja toisena yhteyden. Yhteytenä voimme tässä tapauksessa käyttää NetworkBehaviourista saatua valmista connectionToClient muuttujaa, joka yksilöi pelaajan clientin siihen käyttäjään, joka ajaa koodia. NetworkServer.SpawnWithClientAuthorityä käytetään siis luomaan verkotettu objekti, jolle halutaan määrittää auktoriteetti jollekkin pelin pelaajalle, eikä serverille.

4.3 Remote Procedure Callit (ClientRpc)

Remote Procedure Callit, lyhyemmin ClientRpc:t ovat komentojen vastakohta. Ne eivät lähetä clientiltä serverille tietoa, vaan päinvastoin serveriltä halutuille tai kaikille clienteille (Unity3D p 2019). ClientRpc:t tehdään myös aina omaan metodiinsa, ja niiden yläpuolelle tulee kirjoittaa rivi [ClientRpc]. ClientRpc:n metodin nimi pitää myös alkaa sanalla Rpc, esimerkiksi void RpcChangeHealth(). ClientRpc komentoja vastaavasti voi lähettää vain ja ainoastaan serveri eteenpäin (Unity3D p 2019). Clientit eivät voi keskustella keskenään serveri-clientti arkkitehtuurissa, vaan kaikki tieto menee serverin kautta ja siirtyy sieltä eteenpäin.

```
public int health = 100;

    [Command]
    void CmdChangePlayerHealth(int currentHealth) {
        health = currentHealth;
        Debug.Log("Changed health: " + health);

        //verify if this is ok.

        RpcChangeHealth(currentHealth);
    }

    [ClientRpc]
    void RpcChangeHealth(int currentHealth) {
        health = currentHealth;
    }
```

Kuva 10: ClientRpc käytön esimerkki

Ylläolevassa kuvassa näemme yksinkertaisen esimerkin ClientRpc:n käytöstä. Koodissa pelaajalle on tietty määrä elämäpisteitä ja niitä halutaan vaihtaa. Sanotaan esimerkin vuoksi että pelissä on neljä pelaajaa, joista ensimmäinen pelaaja on serveri että clientti samassa ja loput pelaajat ovat normaaleja clienttejä. Pelaaja numero kaksi lähettää serverille komennon vaihtaa hänen omat elämäpisteensä ja vaihtaa samalla omalle clientilleen elämäpisteet oikeiksi. Serveri vastaanottaa komennon ja vaihtaa pelaaja 2:n elämäpisteet oikeiksi serverillä. Tämän jälkeen siis pelaaja 2:n elämäpisteet ovat oikein kahdella clientillä (1 ja 2), kun taas pelaajat 3 ja 4 eivät tiedä muutoksesta mitään, koska niitä ei ole lähetetty eteenpäin. Tästä syystä tarvitaan ClientRpc:itä, koska ne viestittävät serverillä tapahtuvat muutokset kaikille pelaajille.

Yllä olevassa esimerkissä nähdään miten komentoja ja ClientRpc:itä voidaan käyttää sisäkkäin. Koska komennot lähetetään clientiltä ja suoritetaan aina serverillä, voidaan ClientRpc:n kutsu lisätä suoraan komennon sisälle. Näin saadaan automatisoitua systeemi, jolla pelaaja pyytää serveriltä elämänmuutosta ja serveri lähettää sen samalla eteenpäin kaikille muille käyttäjille.

Komentoihin ja ClientRpc metodeihin voi lisätä argumentteihin muuttujia normaalisti, mutta pitää huomioida se, että vain tiettyjä muuttujatyyppejä voi lähettää verkon yli. Perustyypit kuten int, float, string, vector3 ja quaternion toimivat ilman muutoksia, mutta esimerkiksi itsetehtyjen luokkien lähettäminen muuttujana voi

olla hankalaa, riippuen luokan monimutkaisuudesta. (Unity3D p 2019) Tälläisiä luokkia voi lähettää, mutta ne pitää muuttaa ensin byteiksi. Luokka muutetaan byteiksi, jonka jälkeen voidaan lähettää bytet ja sen jälkeen kasata luokka uusiksi vastaanottamisen jälkeen. Byteiksi muuttajaa ei tarvitse koodata usein itse ellei halua, koska netistä löytyy paljon esimerkkikoodia tämän prosessin hoitamiseen.

4.4 SyncVarit

SyncVar (Synchronized Variable, synkronoitu muuttuja) on keino nopeuttaa yksinkertaisten muuttujien koodaamista verkkopeleissä. Kaikki SyncVarin toiminnot voidaan tehdä myös ClientRpc:itä käyttämällä omin käsin, mutta se on hitaampaa ja vie enemmän tilaa ja aikaa koodata. SyncVar muuttujia voidaan muuttaa vain serverillä ja ne lähettävät automaattisesti muutoksista tiedot kaikille clienteille (Unity3D q 2019). SyncVarin muuttaminen clientillä ei tee mitään.

```
[SyncVar]
public int health = 100;


[Command]
void CmdChangePlayerHealth(int currentHealth) {

    health = currentHealth;
}
```

Kuva 11: SyncVar esimerkki

Ylläoleva esimerkki tekee täysin samat asiat, kuin ClientRpc kohdan esimerkki elämäpisteiden muutoksesta. SyncVar hoitaa vain ClientRpc:n lähettämisen automaattisesti ilman turhaa koodaustyötä. Kuten nähdään, SyncVarien käyttö pienentää huomattavasti tarvittavan koodin määrää. Silti varsinaisesti SyncVarien käyttö ei ole pakollista, selvyyden ja ymmärtämisen vuoksi voidaan koodata kaikki omia ClientRpc:itä käyttäen. SyncVarit silti käyttävät salaisesti ClientRpc komentoja, joten omien ClientRpc:iden tekeminen ei hidasta peliä yhtään, ne ovat vain hitaampia kirjoittaa. SyncVar toimii perustyyppisten muttujien kanssa sekä Unityyn sisäänrakennettujen matemaattisten tyyppien kanssa.

4.5 Hookit

Hookit on valinnainen lisäkohta SyncVareihin. Niillä suoritetaan haluttu metodi aina, kun SyncVariksi määritelty muuttuja muuttaa arvoaan (Unity3D r 2019). Täysin sama asia voidaan tehdä myös käyttäen ClientRpc:tä metodissa ja kutsumalla metodia komennon mukana. Hookit säästävät aikaa tietyissä tilanteissa, mutta yleisesti ottaen ne eivät ole hirveän hyödyllisiä ja yleensä selvyyden kannalta itse suosittelen käyttämään ClienRpc metodeita hookkien tilalla. Koodari voi tässä tapauksessa päättää käyttääkö hookkeja vai ClientRpc:tä, koska molemmat vievät keskimäärin saman verran rivejä ja koodausaikaa.

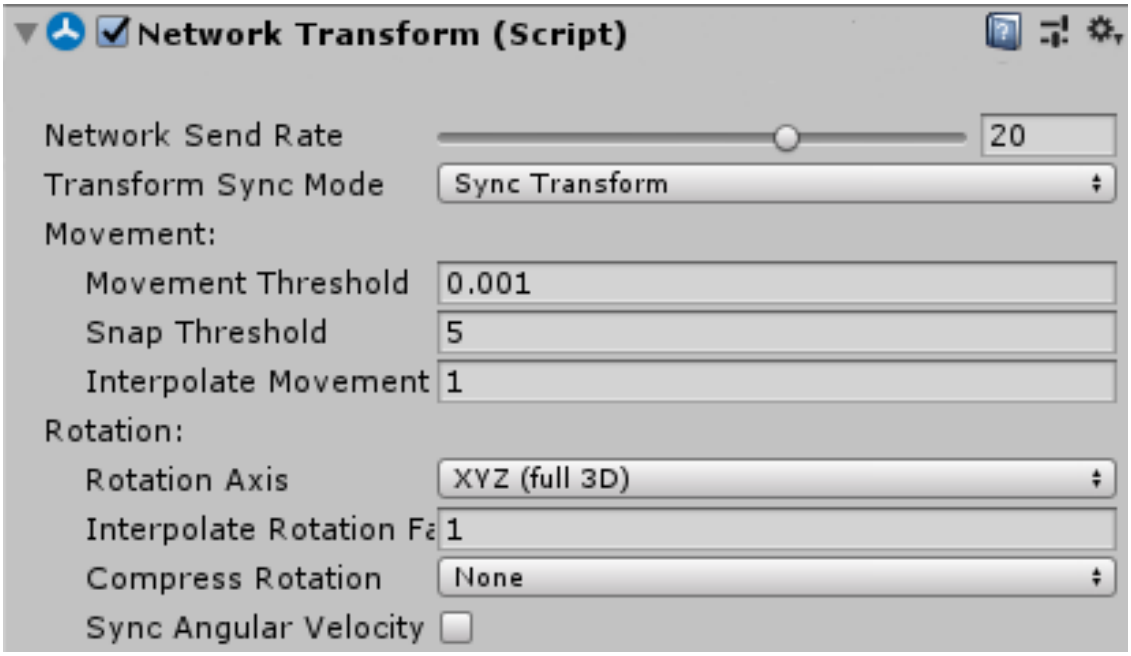
```
[SyncVar(hook="OnHealthChanged")]
public int health = 100;

void OnHealthChanged(int newHealth) {
    Debug.Log("Old health: " + health + " New health: " + newHealth);

    health = newHealth;
}
```

Kuva 12: Hookin käyttö

Kuvassa nähdään hookkien kirjoitusasua hieman tarkemmin. Hookki liitetään SyncVar muuttujaan ja halutun metodin nimi laitetaan lainausmerkkien sisään. Hookkeihin ei tarvitse välttämättä laittaa nimeä, vaan sillä voi käyttää myös delegaatteja. Emme mene sen enempää delegaatteihin tämän opinnäytetyön parissa, koska ne eivät varsinaisesti liity aiheeseen, mutta on hyvä tietää mahdollisuudet.

4.6 NetworkTransform

NetworkTransform on Unity Unetin korkean tason komponentti, jolla voidaan nopeasti prototyypata kappaleiden liikkumista ja kääntymistä verkossa (Unity3D s 2019). Komponentti toimii hyvin vain ja ainoastaan prototyyppaamiseen, koska sen käytössä on muutamia ongelmia, jotka voidaan helposti korjata koodaamalla parempi komponentti valmiiseen peliin. NetworkTransform komponentti lisätään haluttuun liikuteltavaan tai käännettävään objektiin.



Kuva 13: NetworkTransform komponentti

NetworkTransform komponentissa on muutamia säätövaihtoehtoja, mutta mihinkään ei välttämättä tarvitse koskea aloitusasetuksista. Komponentti on hyvä vain nopeaan prototyyppaamiseen, joten aikaa on turha käyttää ylimääräisten säätöjen tekemiseen. NetworkTransformin isoin ongelma on se, että sen interpolaatio toimii hyvin huonosti.

Käytännössä NetworkTransform lähettää vain koordinaatteja verkon yli ja siirtää objektia niihin koordinaatteihin. Objekti liikkuu verkossa sinällään oikein, mutta liike on erittäin tökkivää, vaikka lähettäisi koordinaatteja komponentin maksimimäärän verran, eli 29 kertaa sekunnissa. Network Send Raten säätö maksimiin vie myös enemmän kaistaa ja se on näin ollen erittäin epätehokasta verkon käyttöä.

4.7 NetworkTransformin tekeminen itse

Tässä kappaleessa katsomme hieman yksinkertaista itsetehtyä NetworkTransformin korviketta, jolla saadaan interpolaatio mukaan peliobjekteihin, jolloin ne eivät liiku enää pätkien vaan tasoittavat liikettä lineaarisella interpolaatiolla. Koodia käytetään siinä objektissa, jota halutaan liikuteltavan. Huomioitavaa on se,

että mahdollinen NetworkTransform komponentti pitää poistaa objektista kustomoitua versiota käytettäessä. Koodia katsotaan kahdessa eri osassa, ensin katsotaan tarvittavia apumetodeja ja sitten toiseksi apumetodien käyttöä ja itse koodin toiminnallisuutta.

```
public class NetworkTransformRework : NetworkBehaviour {
    public float posRate = 15;
    public float rotRate = 15;
    public float posThreshold = 0.1f;
    public float rotThreshold = 1f;

    [SyncVar]
    private Vector3 lastPosition;
    [SyncVar]
    private Vector3 lastRotation;

    [Command]
    private void CmdSendPosition(Vector3 pos) {
        lastPosition = pos;
    }

    [Command]
    private void CmdSendRotation(Vector3 rot) {
        lastRotation = rot;
    }

    private void InterpolatePosition() {
        transform.position = Vector3.Lerp(transform.position, lastPosition, Time.deltaTime * posRate);
    }

    private void InterpolateRotation() {
        transform.rotation = Quaternion.Lerp(transform.rotation, Quaternion.Euler(lastRotation), Time.deltaTime * rotRate);
    }

    private bool HasPositionChanged() {
        return Vector3.Distance(transform.position, lastPosition) > posThreshold;
    }

    private bool HasRotationChanged() {
        return Vector3.Distance(transform.localEulerAngles, lastRotation) > rotThreshold;
    }
}
```

Kuva 14: Kustomoidun NetworkTransformin apumetodit

Omaa NetworkTransformia kirjoittaessa tarvitaan muutama tärkeä muuttuja. Muuttujan arvot posRate ja rotRate tarkoittavat sitä, montako kertaa arvoja päivitetään sekunnin aikana. Tämä on siis sama kuin NetworkTransformin Network Send Rate. Muuttujat posThreshold ja rotThreshold määrittävät sen, kuinka paljon peliobjektin tulee liikkua tai kääntyä, ennen kuin päivitetään tieto verkossa eteenpäin. SyncVar muuttujina on viimeinen sijainti sekä viimeinen käännöksen suunta, joita päivitetään serveriltä automaattisesti eteenpäin clienteille, koska kyseessä on SyncVar.

Komennot ovat tässä luokassa helpot, koska niillä lähetetään eteenpäin serverille vain nykyinen positio ja käännöksen suunta. Komentojen jälkeiset kaksi metodia on tarkoitettu Lerpin (Lineaarinen Interpolaatio) käyttöön liikkumisessa ja käännöksissä, jotta liikkeestä saadaan tasaista ja ei pätkivää. Viimeiset kaksi metodia tarkistavat sen, että onko kappale liikkunut tai kääntynyt halutun rajan yli. Tämä

on tärkeä tarkistaa, sillä emme halua missään tapauksessa lähettää ylimääräisiä kutsuja verkon yli, koska se on turhaa verkkoliikennettä ja se kuormittaa peliä paljon.

```
void Update() {
    if (hasAuthority) {
        return;
    }

    InterpolatePosition();
    InterpolateRotation();
}

void FixedUpdate() {
    if (!hasAuthority) {
        return;
    }

    if (HasPositionChanged()) {
        CmdSendPosition(transform.position);
        lastPosition = transform.position;
    }

    if (HasRotationChanged()) {
        CmdSendRotation(transform.localEulerAngles);
        lastRotation = transform.localEulerAngles;
    }
}
```

Kuva 15: Kustomoidun NetworkTransformin toiminnallisuus

Ylläolevassa kuvassa nähdään koodin toiminnallisuuden osa. On tärkeää ensiksi huomata se, että toiminnallisuus on jaettu metodeihin Update ja FixedUpdate. Yksi peliobjekti ei koskaan suorita molempia samanaikaisesti tässä koodissa. Syy siihen on se, että Update metodin koodi halutaan suorittaa vain silloin kun clientillä ei ole auktoriteettiä objektiin ja FixedUpdaten taas silloin kun peliobjekti on oma objektisi eli käyttäjällä on auktoriteetti siihen.

Käytännössä jos kyse on omasta objektista, koodi lähettää serverille tiedon viimeisestä positiosta ja viimeisestä käännössuunnasta, jos sille on tarvetta. Koska objekti on oma, sen ei tarvitse vastaanottaa serveriltä paikkatietoja, koska omat tiedot ovat aina serveriä edellä. Vastaavasti jos objekti ei ole oma, vaan se on jonkin toisen clientin objekti verkon yli, koodi vain vastaanottaa koordinaatteja serveriltä ja yrittää niiden mukaan siirtää objektia oikeaan paikkaan.

Verkkopeleissä on tärkeää aina muistaa, että kaikkien peliin tehtävien asioiden on tunnuttava hyvältä pelaajan näkökulmasta. Tämä on syy miksi esimerkiksi usein liikkuminen tehdään reaaliaikaisissa verkkopeleissä paikallisesti pelaajille, eikä verkon yli. Liikuttelu paikallisesti poistaa täysin viiveen, jolloin pelaajan napin painallukset tekevät halutut asiat heti. Vuoropohjaisissa verkkopeleissä voidaan liikuttelu hoitaa pelkillä komennoilla ja ClientRpc kutsuilla, koska viiveellä ei ole niin paljon merkitystä pelattavuuteen.

5 POHDINTA

Tässä työssä eriteltiin erilaisia verkkotyökaluja, joilla pelin kehittäjä voi halutessaan tehdä verkkopelin. Pelinkehittäjän vastuulla on miettiä, mikä työkalu sopii hänen haluamaansa peliin parhaiten. Kaikki työkalut ovat alan ammattilaisten tekemiä ja niillä pystyy taatusti luomaan verkkopelejä, jotkut vaativat vain enemmän opiskelua, kuin toiset. Pelin tyypin kannalta on tärkeää miettiä käytettävää työkalua etukäteen, koska kesken projektin verkkotyökalun vaihtaminen on erittäin työlästä.

Tässä opinnäytetyössä käytiin läpi myös Unity Unetin käyttöä ja sen toimintaperiaatteita, jotka eroavat huomattavasti yksinpelattavan pelin tekemisestä. Unity Unetin käytön opettaminen pyrittiin pitämään yleispätevänä ja hyödyllisenä. Vaikka päätyisi käyttämään jotain muuta verkkotyökalua, kuin Unity Unetiä, on esimerkit silti varmasti hyödyllisiä lukea. Useimmat verkkotyökalut pohjautuvat samoihin periaatteisiin, jolloin yhden osaaminen yleensä antaa avaimet helposti opetella minkä tahansa työkalun helposti.

Verkkopelien tekemiseen käytetyt teknologiat kehittyvät koko ajan huimaa vauhtia ja on hyvin mahdollista, että jo muutaman vuoden sisällä käytetyimmät verkkotyökalut ovat jo täysin erilaiset, kuin tänäpäivänä. Verkkopelien tekeminen tällä hetkellä on suhteellisen vaikeaa, mutta uskoisin että lähivuosina verkkotyökalut kehittyvät niin pitkälle, että niillä tekeminen muuttuu suoraviivaisemmaksi ja helpommaksi. Verkkotyökalujen kehittyminen tuo lisää tekijöitä verkkopelien pariin, joka taas lisää tutoriaalien ja muiden oppaiden määrää, joka vastaavasti helpottaa opiskelua.

Varsinaisesti tämän aiheen oppimiseen ei riitä pelkästään se, että on lukenut aiheesta ja tietää joitain asioita. Niin kuin yleisestikin ohjelmoinnin parissa työskentelevät tietävät, paras tapa oppia asioita on tehdä ne itse ja oppia siinä sivussa. Verkkopelien tekeminen vaatii ylipäätään paljon ajattelutyötä ja teknistä osaamista, jotta ymmärtää mitä kuuluu lähettää verkon yli ja kuinka usein. Lisäksi verkkopelin tekeminen vaatii tietynlaisen ajattelutavan, koska pitää ajatella mitä useammalla tietokoneen ruudulla näkyy sillä hetkellä eikä vain omalla. Loppujen

lopuksi kaikki on vain kiinni viisaasti tehdystä koodista, jossa serverin ja clienttien kommunikointi toimii selvästi ja mutkattomasti, mitään erikoiskikkoja ei tarvita hyvään pelaajakokemukseen.

LÄHTEET

Darkrift a, 2015. Darkrift Networking original release. Luettu 18.03.2019. https://forum.unity.com/threads/darkrift-fast-and-flexible-cross-platform-networking.320185/

Darkrift b, 2018. Darkrift 2 Networking released. Luettu 18.03.2019. https://forum.unity.com/threads/darkrift-networking-2-the-next-level-of-networking.380800/

Darkrift c, 2018. Darkrift 2 front page. Luettu 18.03.2019. https://darkriftnetworking.com/DarkRift2

Forge a, 2015. Forge Networking released. Luettu 15.03.2019. https://forum.unity.com/threads/new-forge-networking-released.332895/

Forge b, 2017. Forge Networking Remastered released. Luettu 15.03.2019. https://github.com/BeardedManStudios/ForgeNetworkingRemastered

Forge c, 2017. Forge Networking. Luettu 15.03.2019. https://developers.forge-powered.com/

Juhl, E., 2014. Announcing UNET new multiplayer technology. Luettu 05.12.2018. https://blogs.unity3d.com/2014/05/12/announcing-unet-new-unity-multiplayer-technology/

Margaret Rouse, Techtarget, 2018. UDP. Luettu 12.12.2018. https://searchnetworking.techtarget.com/definition/UDP-User-Datagram-Protocol

Netscout, 2014. UDP. Luettu 12.12.2018. https://enterprise.netscout.com/edge/tech-tips/difference-between-tcp-and-udp

Photon a, 2019. Photon Unity Networking vs Photon Bolt. Luettu 17.12.2018. https://doc.photonengine.com/en-us/pun/v2/reference/pun-vs-bolt

Photon b, 2019. Photon Unity Networking. Luettu 16.12.2018. https://www.photonengine.com/pun

Photon c, 2019. Photon Bolt. Luettu 17.12.2018. https://www.photonengine.com/bolt

StackExchange, 2017. How is Forge able to provide unlimited CCU? Luettu 16.03.2019. https://gamedev.stackexchange.com/questions/143464/how-is-forge-networking-able-to-provide-unlimited-ccu

TechnicalTerms, 2017. Latency. Luettu 05.12.2018. https://techterms.com/definition/latency

Unity3D a, 2019. Public Relations of Unity. Luettu 06.12.2018. https://unity3d.com/public-relations

Unity3D b, 2019. Store page. Luettu 06.12.2018. https://store.unity.com/

Unity3D c, 2019. Public relations of Unity. Luettu 08.12.2018. https://unity3d.com/unity/features/multiplatform

Unity3D d, 2009. Unity free version released. Luettu 08.12.2018. https://unity3d.com/company/public-relations/news/unity2.6-press

Unity3D e, 2014. Unity blogpost about scripts. Luettu 09.12.2018. https://blogs.unity3d.com/2014/09/03/documentation-unity-scripting-languages-and-you/

Unity3D f, 2019. Unity services. Luettu 09.12.2018. https://unity3d.com/unity/features/multiplayer

Unity3D g, 2019. Multiplayer overview. Luettu 14.12.2018. https://docs.unity3d.com/Manual/UNetOverview.html

Unity3D h, 2019. Unet using HLAPI. Luettu 14.12.2018. https://docs.unity3d.com/Manual/UNetUsingHLAPI.html

Unity3D i, 2019. Unet HLAPI concepts. Luettu 14.12.2018. https://docs.unity3d.com/Manual/UNetConcepts.html

Unity3D j, 2015. Networking solutions review. Luettu 21.03.2019. https://forum.unity.com/threads/multiplayer-networking-solutions-review-photon-ulink-darkrift-forge-and-playfab.294852/

Unity3D k, 2019. NetworkManager komponentti. Luettu 29.03.2019. https://docs.unity3d.com/ScriptReference/Networking.NetworkManager.html

Unity3D l, 2019. NetworkIdentity komponentti. Luettu 01.04.2019. https://docs.unity3d.com/Manual/class-NetworkIdentity.html

Unity3D m, 2019. Unet scene objects. Luettu 01.04.2019. https://docs.unity3d.com/Manual/UNetSceneObjects.html

Unity3D n, 2019. NetworkManagerHUD komponentti. Luettu 03.04.2019. https://docs.unity3d.com/ScriptReference/Networking.NetworkManagerHUD.html

Unity3D o, 2019. Command attribuutti. Luettu 05.04.2019. https://docs.unity3d.com/ScriptReference/Networking.CommandAttribute.html

Unity3D p, 2019. ClientRpc attribuutti. Luettu 06.04.2019. https://docs.unity3d.com/ScriptReference/Networking.ClientRpcAttribute.html

Unity3D q, 2019. SyncVar attribuutti. Luettu 07.04.2019. https://docs.unity3d.com/ScriptReference/Networking.SyncVarAttribute.html

Unity3D r, 2019. SyncVarin attribuutti hook. Luettu 07.04.2019. https://docs.unity3d.com/ScriptReference/Networking.SyncVarAttribute-hook.html

Unity3D s, 2019. NetworkTransform komponentti. Luettu 09.04.2019. https://docs.unity3d.com/Manual/class-NetworkTransform.html

Wikipedia a, 2019. Unity game engine. Luettu 06.12.2018. https://en.wikipedia.org/wiki/Unity_(game_engine)

Wikipedia b, 2019. UDP. Luettu 12.12.2018. https://fi.wikipedia.org/wiki/UDP

Wikipedia c, 2019. Client-server model. Luettu 15.12.2018. https://en.wikipedia.org/wiki/Client%E2%80%93server_model

Youtube, 2017. Photon vs Unet – multiplayer architecture explained. Katsottu 16.12.2018. https://www.youtube.com/watch?v=Y1my5bKhKJY