KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

Aki Väätäinen

ASYMMETRINEN PELINKEHITYS PC:N JA MOBIILILAITTEIDEN VÄLILLÄ

Opinnäytetyö
Marraskuu 2014

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Marraskuu 2014
Tietojenkäsittelyn koulutusohjelma

Karjalankatu 3
80200 JOENSUU
puh. 013 260 600

Tekijä
Aki Väätäinen

Nimeke
Asymmetrinen pelinkehitys PC:n ja mobiililaitteiden välillä

Toimeksiantaja
Mental Moustache Oy

Tiivistelmä

Opinnäytetyö tutkii asymmetristen pelien kehitystä. Ne ovat moninpelejä, joissa pelaajien roolit vaihtelevat pelitilanteesta tai laitteistosta riippuen. Työ vastaa seuraaviin tutkimusongelmiin: Miten asymmetrinen peli voidaan toteuttaa teknisesti? Miten asymmetrisyys soveltuu PC-alustalle? Miten mobiililaitteita voidaan hyödyntää PC-pelaamisessa?

Mobiililaitteiden hyödyntäminen ja erilaisten roolien selvittäminen tapahtuu Wii U -konsolin pelejä analysoimalla. Teknistä toteutusta, asymmetrisiä pelejä ja niiden yhteyttä PC:hen käsitellään ensin teoreettisesti ja lopuksi toteutetaan peliprototyyppi Unity-pelimoottoria hyödyntäen. Mobiililaitteet tarkoittavat tässä asiayhteydessä älypuhelimia sekä tabletteja, sillä toteutus on rajoitettu näihin laitteisiin.

Asymmetrisyys on mahdollista toteuttaa verkko-ohjelmoinnin avulla. Työn tuloksena on prototyyppi. Peli on reaaliaikaisesti yhtenäinen, mutta sitä pelataan alustan määrittelemän roolin mukaan. Mobiililaitteita voidaan hyödyntää monella tapaa: niitä voidaan käyttää esimerkiksi ylimääräisenä ohjaimena (kosketusnäyttö, kiihtyvyysanturi), tietojen näyttämiseen tai toisena näyttönä. Asymmetrisyys sopii PC:lle erittäin hyvin, sillä yleisesti saatavilla olevia mobiililaitteita voidaan käyttää toteutuksessa. Lisäksi pelitilojen synkronoinnilla toteutettuna yhteys oli prototyypissä toimiva, ja se tukee useampaa pelaajaa kerralla.

Mobiililaitteita ja asymmetrisyyttä käyttävillä peleillä on potentiaalia, mutta täysin ongelmattomia ne eivät ole. Vielä voitaisiin tutkia muiden verkko-ohjelmointiympäristöjen, mobiili- ja lisälaitteiden hyödyntämistä ja videokuvan lähettämistä asymmetristen pelien toteuttamiseksi.

Kieli	Sivuja 100
suomi	Liitteet 1

Asiasanat
moninpelit, peliohjelmointi, verkko-ohjelmointi, Unity

THESIS
November 2014
Degree Programme in Business Information Technology

Karjalankatu 3
FI 80200 JOENSUU
FINLAND
Tel. 013 260 600

Author
Aki Väätäinen

Title
Asymmetric Game Development between PC and Mobile Devices

Commissioned by
Mental Moustache Ltd.

Abstract

The thesis studies the development of asymmetric games. They are multiplayer games, in which the players' roles vary depending on the game situation or hardware. This thesis answers the following research questions: How can an asymmetric game be implemented? How well does asymmetry work on the PC platform? How can mobile devices be utilized in PC-gaming?

Utilizing mobile devices and examining different roles happens by analyzing the games of the Wii U console. Implementation, asymmetric games and their connection to the PC platform is discussed first theoretically and finally a game prototype is implemented using the Unity game engine. Mobile devices in this context mean smart phones and tablets, since the implementation has been restricted to these devices.

It is possible to implement asymmetry by using network programming. The result of this thesis is a prototype. The game is coherent in real-time, but it is played by the roles determined by each platform. Mobile devices can be utilized in many ways: they can be used as an additional controller (touch screen, accelerometer), displaying information or as a second screen. Asymmetry works on the PC very well, because mobile devices are widely available and they can be used in the implementation. In addition, the connection between devices worked by synchronizing game states. It supports multiple players at once.

Games that utilize asymmetry and mobile devices have potential, but they are not completely without problems. Additionally using third-party networking solutions, streaming video, mobile devices and other hardware could be examined in implementing asymmetric games.

Language	Pages 100
Finnish	Appendices 1

Keywords
multiplayer games, game programming, network programming, Unity

Sisältö

1 Johdanto

Asymmetrisyys on peleissä uusi ilmiö. Sillä tarkoitetaan moninpeliä, jossa jokainen pelaaja tekee hieman eri asioita laitteesta tai tilanteesta riippuen. Wii U -konsoli julkaistiin vuoden 2012 lopulla, mikä sai aikaan uudenlaisen tavan kehittää pelejä. Konsolin GamePad-ohjaimessa on muiden painikkeiden lisäksi kosketusnäyttö, joka mahdollistaa usean pelitilan näyttämisen yhtä aikaa. Useampi pelaaja voi pelata yhtä peliä samaan aikaan eri näytöillä ja eri tilanteissa. Se on perusta asymmetriselle pelaamiselle. (Farrington 2012.)

Vaikka asymmetrisyyttä on kokeiltu muillakin alustoilla, PC:llä se on melko tuntematon ilmiö. Opinnäytetyölle rajattiin seuraavat tutkimusongelmat: Miten asymmetrinen peli voidaan toteuttaa teknisesti? Miten asymmetrisyys soveltuu PC-alustalle? Miten mobiililaitteita voidaan hyödyntää PC-pelaamisessa? Taustatutkimukseen kuuluu asymmetristen pelien, alustojen ja roolien analysointi; niiden tarkastelu on erityisen tärkeää kahden jälkimmäisen kysymyksen kannalta.

Mobiililaitteilla tarkoitetaan liikkuvuuteen suunniteltuja kevyitä laitteita (Technopedia 2014). Tämän työn asiayhteydessä mobiililaitteilla tarkoitetaan käytännössä älypuhelinta ja tablet-tietokonetta. Älypuhelimet ovat matkapuhelimia, jotka pystyvät samankaltaisiin toiminnallisuuksiin kuin tietokone (PCMag 2014). Niiden hallintaan käytetään kosketusnäyttöä. Tabletit ovat samankaltaisia, mutta näyttö on suurempi ja niistä puuttuu tavallisen puhelimen ominaisuudet.

Tämän työn perustana on toimeksianto Mental Moustache Oy:ltä, joka halusi tutkittavan aihetta. Heidän motiiveinaan tälle toimeksiannolle on pysyä mukana kehityksessä sekä toteuttaa potentiaalisia ideoita prototyyppitasolla. Asymmetrisyys voisi myös sopia hyvin yhteen Oculus Rift -virtuaalitodellisuuslaseihin. Pelistä toteutetaan prototyyppi, joka sisältää asymmetristä pelaamista. Kokemuksen tarkoitus on olla samantapainen kuin Wii U -konsolilla, mutta laitteet ovat erilaiset. Verkko-ohjelmointi liittyy pelin toteutukseen läheisesti, sillä yhteys luodaan PC:n ja mobiililaitteiden välille. Peliprototyyppi on kokeilu, jolla on mahdol-

lista nähdä, miten hyvin asymmetrisyys toimii PC:llä. Itselläni ei ollut aikaisempaa kokemusta asymmetrisistä peleistä; en ollut oikeastaan pelannut niitä tai tietänyt aiheesta ennalta mitään. Olin tutustunut aiemmin Wii U:n konsoliin sekä GamePadiin pelkästään julkaisutietojen perusteella.

Prototyypin aiheena on ammuskelupeli. PC-pelaajan rooli on ensimmäisen persoonan räiskintä (FPS), kun peliin mukaan liittyvillä mobiilipelaajilla se on ylhäältä kuvattu ammuskelu (top-down shooter). PC-pelaajan täytyy kulkea pelissä tietty reitti, jonka varrella on vihollisten pesiä. Pelaajan täytyy tuhota ne päästäkseen etenemään. Mobiilipelaajat voivat liittyä peliin avustamaan pelaajaa.

Peli toteutetaan Unity-pelimoottorilla[1]. Se mahdollistaa pelinkehityksen monelle alustalle samanaikaisesti (Unity Technologies 2014a), mikä on tässä työssä erittäin tärkeää. Lisäksi Unity tarjoaa RakNetiin perustuvat helppokäyttöiset verkko-ohjelmointikirjastot[2], joita voidaan hyötykäyttää PC:n ja mobiililaitteiden pelitilanteiden synkronoimisessa (RakNet 2014a). Nämä seikat mahdollistavat prototyypille luonteisenomaisen nopeatahtisen kehityksen.

Robert S. Speck (2013) on tutkinut työssään asymmetrisyyden kehittämistä erilaisten laitteiden välille. Hänen työnsä keskittyy esimerkiksi mobiililaitteiden vertailuun, toteutuksen työkaluihin ja alustoihin sekä itse pelikokemuksen suunnitteluun vaikuttaviin asioihin (esimerkiksi suorituskyky, käyttöliittymät ja syötteiden toiminta). Myös hän päätyi käyttämään prototyypissään Unity-pelimoottoria. Työ ei kuitenkaan kerro paljon pelin teknisestä toteutuksesta yksityiskohtaisemmin tai analysoi asymmetrisiä pelejä. Opinnäytetyöni pyrkii tutkimaan asymmetrisyyttä erilaisesta näkökulmasta, erityisesti teknisen toteutuksen ja hyötykäyttömahdollisuuksien kannalta.

Toisessa luvussa kuvataan mobiililaitteiden ja mobiilipelaamisen tilannetta yleisesti: kuinka yleisiä mobiililaitteet ovat ja miten paljon pelejä pelataan?

1 Pelimoottorilla on mahdollista luoda pelejä nopeammin hyödyntämällä valmista pohjaa, se sisältää useimmiten muun muassa toiminnallisuudet 3D-grafiikan piirtämiseksi, äänten käsittelyn, syötteiden hallinnan sekä fysiikan ja törmäysten määrittelyn.

2 Kirjastot sisältävät valmiita toiminnallisuuksia, joita voidaan käyttää ohjelmoinnissa.

Kolmannessa luvussa selvitetään, mitkä alustat hyödyntävät asymmetrisyyttä ja miten se tapahtuu. Luvussa keskitytään analysoimaan esimerkiksi Wii U -konsolin GamePadin toimintaa sekä lyhyesti asymmetristen pelien ja laitteiston historiaa.

Neljännessä luvussa analysoidaan Wii U -konsolin pelejä tarkemmin, minkä perusteella voidaan pohtia GamePadin ja mobiililaitteen hyötykäyttömahdollisuuksia. Esimerkeissä käsitellään myös lyhyesti asymmetrisiä rooleja.

Viidennessä luvussa asymmetrisyyttä ja mobiililaitteen hyödyntämistä käsitellään PC:n näkökulmasta. Mikä on PC:n ja mobiililaitteiden suhde? Käytetäänkö mobiililaitteita PC-peleissä? Millaisia asymmetrisiä kokemuksia tällä alustalla tarjotaan? Lisäksi tarkastellaan PC:lle kehitettyä laitteistoa asymmetrisyyden näkökulmasta.

Kuudennessa luvussa tarkastellaan asymmetrisen pelin toteutuksen menetelmiä. Luvussa tarkastellaan Unity-pelimoottorin verkko-ominaisuuksien toiminta, miten peli jaetaan ja asymmetrisyys toteutetaan eri laitteiden välillä.

Seitsemännessä luvussa käsitellään peliprototyypin sekä asymmetrisyyden toteuttamista käytännön tasolla.

Viimeisessä luvussa kritisoidaan asymmetrisyyttä, esitellään työn tulokset sekä pohditaan mahdollisia ajatuksia jatkokehitystä varten.

2 Mobiililaitteiden tilanne pelinkehityksessä

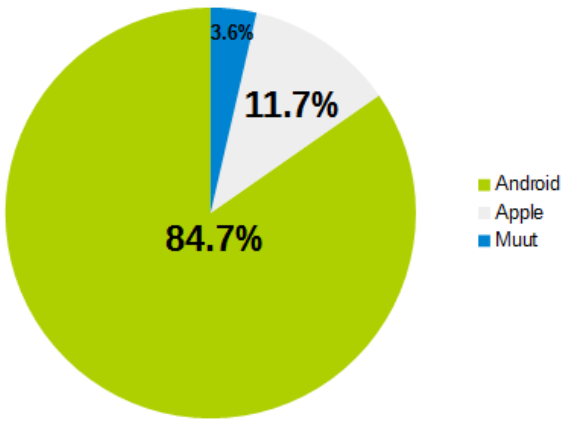
Mobiililaitteista on kehittynyt viime vuosina varsin arkipäiväisesti käytettyjä laitteita. Noin 80 %:lla 16–34-vuotiaista suomalaisista oli käytössään älypuhelin (Tilastokeskus 2013). Tässä ikäryhmässä myös videopelien pelaaminen on yleistä (ESA 2013, 2). Tablet-tietokoneet ovat yleistyneet huomattavasti ottaen huomioon, että vuonna 2011 sellainen oli vain 4 %:ssa suomalaisista kotitalouksista (Tilastokeskus 2011, 7). Kahdessa vuodessa tablet-tietokoneiden määrä nousi

20 %:iin (Tilastokeskus 2013). Koska mobiililaitteet (erityisesti älypuhelimet) ovat yleisiä, niiden hyötykäyttöä on hyvä pohtia pelinkehityksen näkökulmasta.

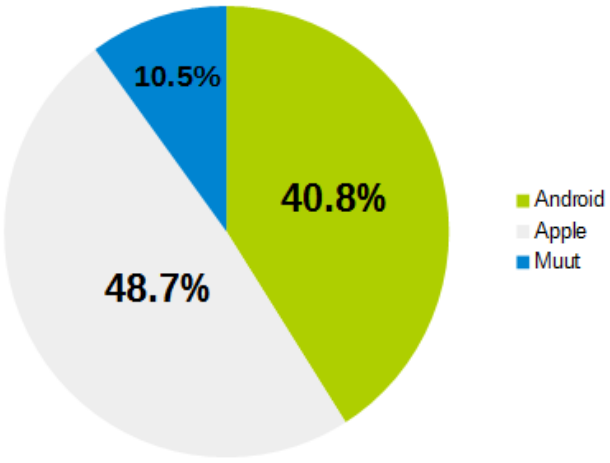
Älypuhelimet ovat olleet varteenotettava alusta peleille jo vuosia. Vuonna 2013 mobiilipelaajia oli maailmalla yli 909 miljoonaa, mikä kertoo pelikäytön valtavasta suuruudesta maailmalla (Statista 2014). Erityisesti nuoret pelaavat paljon mobiilipelejä. Puzzle- ja niin sanotut casual-pelit ovat mobiilipelaajien suosiossa. (ESA 2014). Älypuhelimille suunnatut pelit ovat suositumpia kuin tablet-pelit. Pelejä pelataan pääasiassa kotona huolimatta siitä, että laitteet ovat kannettavia. (Statista 2014.)

Pelimarkkinat ovat suuret, jopa yli 90 miljardia dollaria vuonna 2013. Tästä noin 13,2 miljardia dollaria kuului mobiilipeleille. (GameSpot 2013.) Mobiilimarkkinat kasvavat nopeaa tahtia, niiden arvon odotetaan jopa kaksinkertaistuvan vuoteen 2017 mennessä. Erityisesti Aasian markkinat ovat suuret. (Newzoo 2014.)

Vuoden 2014 toisella vuosineljänneksellä Android oli suosituin älypuhelinalusta. Applen tuotteiden suosio on laskenut hieman. Windows Phonen markkinaosuus on vähäistä. (IDC 2014.) Osuudet on kuvattu kuviossa 1. Android-pohjaiset laitteet ovat halpoja, joten suosio älypuhelinmarkkinoilla ei ole sen perusteella yllätys. Jos huomioidaan myös tabletit, Applella on ollut suurin markkinaosuus (NetMarketShare 2014). Osuudet on kuvattu tarkemmin kuviossa 2. Pelejä kehitettäessä mobiililaitteille on tarpeellista huomioida eri laitealustat ja niiden suosio.



Kuvio 1. Suosituimmat älypuhelinalustat (IDC 2014).



Kuvio 2. Mobiilialustojen kokonaismarkkinatilanne tammikuu-syyskuu (NetMarketShare 2014).

Mobiililaitteiden yleisyys ja suosio huomioon ottaen niitä voitaisiin käyttää PC:n kanssa yhtenäisesti, mikä avaa paljon uusia mahdollisuuksia mobiililaitteiden hyödyntämiseen. Tästä johtuen asymmetrisyyden toteuttaminen PC:n ja mobiililaitteiden välille ja mobiililaitteiden hyödyntäminen ovat erittäin ajankohtaisia. Asymmetristen pelien pelaaminen esimerkiksi omalla älypuhelimella ei olisi suuri este. Tämän vuoksi on hyvä pohtia, miten mobiililaitteita voitaisiin hyödyntää peleissä. Niiden avulla olisi mahdollista tehdä uudenlaisia pelejä, millä on erityistä arvoa pelinkehittäjille. Mobiilimarkkinoiden kasvu kertoo kehityssuunnasta, jossa älypuhelimia käytetään yhä yleisemmin pelaamiseen.

3 Asymmetrisyyttä hyödyntävät alustat

Ensin on hyvä tarkastella mitkä alustat hyödyntävät asymmetrisyyttä. Miten näitä laitteita käytetään ja miten ne liittyvät itse alustaan? Tässä osiossa keskitytään erityisesti pelikonsoleihin. Sen jälkeen on mahdollista verrata muiden alustojen tarjoamaa kokemusta PC:n ja mobiililaitteiden välillä.

3.1 Asymmetrisyyden tarkempi määritelmä

Asymmetrisen pelin perustana on pelaajille määritellyt roolit, jotka ovat erilaiset riippuen tilanteesta tai laitteistosta. Käytännössä se on synonyymi myös asymmetriselle moninpelille. (Giant Bomb 2014; Snyder 2014.) Periaatteessa asymmetrisyys vaatii peliltä hyvin vähän: yksi varhaisimmista asymmetrisistä peleistä on Nintendon Duck Hunt (1984), jossa moninpelissä yksi pelaajista voi ampua sorsia ruudulla, kun toinen liikuttaa niitä (IGN 2014a). Esimerkiksi strategiapelissä kaksi pelaajaa voisi puolustautua yhtä hyökkääjää vastaan. Myös yksinpeleissä voi olla elementtejä asymmetrisistä peleistä, sillä laitteita voidaan hyödyntää niin yksin- kuin moninpeleissäkin. (Giant Bomb 2014.) Jos peli on asymmetrinen, voidaan pelaajan näkökulmasta puhua asymmetrisestä pelikokemuksesta.

Millaisia laitteita asymmetrisissä peleissä voidaan hyödyntää? Esimerkiksi toinen näyttö, liikeohjauksella toimiva kamera tai jokin muu peliohjain voivat toimia sellaisina laitteina. PC:n kohdalla asymmetrinen kokemus voitaisiin luoda esimerkiksi älypuhelimen tai tablet-tietokoneen avulla. Tarkoitus on joka tapauksessa saada aikaan eräänlainen moninpelikokemus, jossa pelaajat voivat tehdä vähän eri asioita riippuen heille asetetusta laitteistosta ja pelitilanteesta.

3.2 Laitteiston hyödyntämisen historia

Laitteiston hyödyntäminen asymmetrisen pelikokemuksen aikaansaamiseksi ei tarkalleen ottaen ole täysin uusi idea, sillä Nintendo on hyödyntänyt laitteiden välistä kommunikaatiota aiemminkin. Erityisesti vuonna 2001 julkaistu konsoli GameCube (CNN 2001) mahdollisti jossain määrin asymmetrisen pelikokemuksen. Tähän aikaan julkaistiin myös Nintendon uusin käsikonsoli (IGN CA 2000) eli Game Boy Advance (kuva 1). Erityisen linkkikaapelin avulla (Game Boy Advance Link Cable) on mahdollista liittää käsikonsoli GameCubeen, jolloin laitteiden välillä on kaksisuuntainen yhteys. Se mahdollistaa muun muassa tiedon jakamisen laitteiden välillä ja moninpelin. (Nintendo 2014a.) Liitäntä konsolin ja laitteen välillä on siis fyysinen kaapeli, joka kiinnitetään peliohjainporttiin. Itse käsikonsolia voidaan myös käyttää peliohjaimena. (Nintendo 2014b.)

Ennen GameCubea Nintendo 64 -konsolilla oli mahdollista siirtää tietoa Game-boy-peleistä konsolipeleihin Transfer Pakin avulla (Nintendo 2014c). Myöhem-min Nintendo DS -käsikonsolin pystyi yhdistämään langattomasti Wii-konsoliin ja esimerkiksi lataamaan pelejä (Ebay 2014). Laitteiden välinen kommunikaatio on asymmetrisissä peleissä tärkeää, jos laitteita halutaan hyödyntää.

Kuva 1. Game Boy Advance -käsikonsoli (Evan-Amos 2011).

Game Boy Advancen hyödyntäminen peleissä

Jopa 58 GameCuben peliä hyödynsi laitteiden välistä yhteyttä. Esimerkiksi Final Fantasy Crystal Chronicles -pelissä oli mahdollista selata pelaajan tavaraluette-loa ja valikoita käsikonsolin näytöllä. Se oli siis kokonaan erillinen näkymä tele-visioruudusta. (CGRundertow 2012.) Toisen näytön käyttäminen johti ilmeisesti myöhemmin uusiin innovaatioihin, sillä Nintendo DS -käsikonsolissa oli kaksi näyttöä yhdessä laitteessa. Samaa ideaa on hyödynnetty myöhemmin myös Nintendo Wii U -konsolilla.

Pac-Man Vs -pelissä oli jo kokonainen asymmetrinen pelikokemus, koska pe-laajille on määritelty eri roolit käytetyn laitteen perusteella. Käsikonsolin näytöllä yksi pelaa tavanomaista Pac-Man-peliä, kun taas jopa kolme muuta ihmispelaa-jaa voi pelata GameCubella haamun roolissa. He koettavat syödä pelaajaa jah-taamalla häntä sokkelossa (kuva 2). (CGRundertow 2012.)

Kuva 2. Pac-Man Vs. Yksi Nintendon varhaisista laitteistoa hyödyntävistä asymmetrisistä peleistä (MobyGames 2008).

Tingle tuner on The Legend of Zelda: The Wind Waker -pelissä saatava tavara. Sillä voi yhdistää GameCuben pelin aiemmin kuvatun linkkiyhteyden avulla Game Boy Advanceen. (CGRundertow 2012.) Käsikonsolin näytölle saa näkyviin esimerkiksi kartan ja tietoja pelialueesta. Sillä voi lisäksi ampua pommeja ja etsiä tavarakätköjä. (Zeldapedia 2014.)

3.3 Wii U (GamePad)

Wii U on uusin Nintendon kehittämä konsoli, joka julkaistiin 18. päivä marraskuuta 2012. Suuri uudistus konsolissa edeltäjäänsä verrattuna on GamePad (kuva 3), joka on erilainen peliohjain. Siinä on 6.2 tuuman kosketusnäyttö, jolla on mahdollista saada näkyviin kuva pelattavasta pelistä yhtä aikaa televisioruudun kanssa. Lisäksi GamePadissa on kaksi analogista tattiohjainta, tavallinen ristiohjain, fyysisiä näppäimiä, kamera, mikrofoni, kaiuttimet sekä sensorit liikkeen tunnistukselle eli gyroskooppi ja kiihtyvyysanturi. (Nintendo 2014d.) Fyysisiä painikkeita lukuun ottamatta se muistuttaa hyvin paljon tablet-tietokonetta tai

Nintendo 3DS -käsikonsolin alaosaa. Wii U tukee yhtä GamePadia kerrallaan, joskin on mahdollista, että kahta ohjainta voi käyttää tulevaisuudessa (VG24/7 2014).

Kuva 3. GamePad-ohjain ja Wii U -konsoli (Takimata 2012).

Miten Wii U:n GamePad toimii? Langattoman yhteyden ja Nintendon ohjelmiston avulla konsolista voidaan lähettää kaksisuuntainen kuva laitteiden välillä. Kommunikaatio perustuu Broadcomin Wi-Fi Miracast -teknologiaan ja vertaisverkkoon. Tarkalleen ottaen yhteys on erityismuoto 802.11N langattomasta verkosta (WLAN). Yhteys on suunniteltu toimimaan vähäisellä viiveellä ja korkealla videonlaadulla sekä kestämään häiriösignaaleja. Esimerkiksi radiosignaalit voivat vaikuttaa yhteyteen. Suositeltu käyttöetäisyys on 8 metriä, mutta testiolosuhteissa vastaava luku on jopa 30 metriä. GamePadin NFC-teknologia (Near Field Communication) mahdollistaisi kommunikoinnin muidenkin sitä käyttävien laitteiden kanssa. (Crecente 2012.)

GamePad avaa paljon mahdollisuuksia alustalle suunniteltaville peleille. Lisäksi ideaa asymmetrisestä peleistä voidaan hyödyntää muuallakin. Yhteistoimin pelattavia moninpelejä kutsutaan co-op-peleiksi. Pelaajat ovat samassa joukkueessa ja heillä on jokin yhteinen tavoite. (Rybka 2014.) Perinteisesti co-op-pelejä pelataan Internetin välityksellä, lähiverkossa eri tietokoneilla tai samalla

tietokoneella tai konsolilla, jonka ruutu on jaettu kahtia. Useammalla näytöllä on mahdollista luoda uudenlaisia co-op-kokemuksia, koska ruutua ei tarvitse jakaa kahtia. Pelaajat voivat olla eri tilanteissa, ja heillä voi olla kokonaan erilaiset roolit. Co-op on tietysti mahdollista toteuttaa tavanomaisin keinoin. Siispä ylimääräisen laitteen lisääminen ei välttämättä tuo tarpeeksi lisäarvoa tavanomaiseen yhteispelikokemukseen verrattuna.

3.4 Playstation (Remote Play)

Myös Sony on käyttänyt muita laitteita konsoliensa yhteydessä. Tarkemmin ottaen vuonna 2006 julkaistu Playstation 3 (GameFAQs 2014) sisältää Remote Play -toiminnon. Se on yhteensopiva Playstation Portable -käsikonsolin (kuva 4) kanssa ja myöhemmin myös uuden PSP Vitan kanssa. Alustalle kehitettyjen pelien täytyi kuitenkin erikseen tukea Remote Play -toimintoa. Ikävä kyllä kovin moni peli ei näin tehnyt. (Unbox Therapy 2011.) Laitteen avulla on mahdollista hallita PS3-konsolia sekä pelata pelejä PSP:n ruudulta. Toisin sanoen pelaaja voi vastaanottaa kuvaa laitteesta ja lähettää näppäinpainalluksia konsolille. Remote Play käyttää joko PS3-konsolin tai langattoman päätepisteen verkkoyhteyttä. (Sony Computer Entertainment America 2014.)

Kuva 4. Sony PSP -käsikonsoli (Edward 2005).

Myös uusi Playstation 4 -konsoli tukee Remote Play -toimintoa, mutta erityinen uudistus on sen tuki myös älypuhelimille. Silloin erillistä laitetta ei välttämättä

tarvita hyödyntääkseen tätä ominaisuutta. (CNET 2014.) Tuki on tosin rajoitettu vain tiettyihin Sonyn Xperia-malleihin (TechRadar 2014).

Miten Remote Play toimii? Ominaisuus on suunniteltu PS4-konsolin artikkitehtuuriin Gaikai-teknologialla. Sony käyttää omaa h.264 video -enkooderia muuttaakseen Playstation-konsolin kuvan pienemmäksi resoluutioksi. Kuva lähetetään langattoman verkkoyhteyden avulla. PSP Vita ottaa lähetetyn tiedon vastaan ja muuttaa sen laitteelle sopivaan muotoon. Yhteys on suunniteltu nopeaksi ja saumattomaksi. Uudemmassa Remote Playn versiossa pelinkehittäjät voivat ottaa ominaisuuden helposti käyttöön, jotta aiemmin esiintyneiltä ongelmilta vältyttäisiin. Laitteistoresurssit olivat isompi ongelma kehittäjän näkökulmasta Playstation 3:lla, mutta uuden laitteiston ja arkkitehtuurisuunnittelun ansiosta Remote Play ei aiheuta niin suurta painolastia konsolille. (What Is Playstation 4 2014.) Remote Play ei kuitenkaan ilmeisesti pyri asymmetriseen pelikokemuseen Wii U:n tapaan.

3.5 Xbox (SmartGlass)

Toinen yritys mobiililaitteiden ja konsolien integroimiseksi on Microsoftin SmartGlass-teknologia vuodelta 2012 (Pope 2014). Kuten PlayStation 4:n Remote Playn kanssa on mahdollista käyttää älypuhelinta, Xbox 360 ja One tukevat vastaavaa ominaisuutta. Ohjelman voi ladata mille tahansa tuetulle mobiililaitteelle. SmartGlassia voidaan käyttää langattomana kosketusnäytöllisenä ohjaimena konsolia varten. Se tarjoaa monia mediatoiminnallisuuksia esimerkiksi elokuvien katsomiseen, musiikin kuunteluun ja Internetin selaamiseen. Näytöllä voi esimerkiksi näkyä tietoja toistetusta mediasta. Tämän lisäksi sitä voidaan käyttää peleissä toisena näyttönä. Esimerkiksi Forza Horizon -pelissä omalla mobiililaitteen näytöllä voidaan näyttää kartta (kuva 5). (TechRadar 2013.) Microsoft on parantanut SmartGlassin toimintaa Xbox One -konsolille: se on paljon nopeampi ja tukee jopa 16 yhtäaikaista laitetta. SmartGlass sisältää myös sosiaalisia ominaisuuksia. (IGN 2013.)

Kuva 5. SmartGlassia hyödyntävä karttanäkymä Forza Horizon -pelissä (Zheng 2012).

4 Pelianalyyseja (Wii U)

Tässä osiossa analysoidaan joitakin WII U -konsolin pelejä ja GamePadin käyttöä. Pelit on valittu siten, että ne havainnollistavat asymmetrisyyttä peleissä sekä GamePadin käyttöä mahdollisimman monipuolisesti. Ne ovat esimerkkejä. Pelien skaala on varsin laaja: Mukana on yhteistoiminnallisia pelejä, kilpailullisia pelejä sekä yksinpelejä. Ne edustavat montaa eri pelien lajityyppiä, muun muassa tasohyppely-, seikkailu, taistelu-, toiminta- sekä ajopelejä. Mukana on paljon toisistaan eroavia peli-ideoita, joskin myös yhteneväisyyksiä GamePadin käytössä alkaa näkyä, mikä kertoo esimerkkien laajuudesta. Asymmetrisyyden laajuuden vuoksi ei kaikkia hyödyntämismahdollisuuksia voida käsitellä; osa niistä saattaa olla vielä keksittävissä. Esimerkkien avulla voidaan havaita miten laitteistoa voidaan hyödyntää pelien tukena ja millaisia rooleja asymmetrisissä peleissä on. Yksi pääpaino tässä osiossa on co-op-moninpeleillä, sillä yhteistoiminnallisuus liittyy usein läheisesti asymmetrisiin peleihin.

4.1 New Super Mario Bros U / Luigi U

New Super Mario Bros. U on Nintendon uusi sivulta kuvattu tasohyppely, joka on modernisoitu versio klassisesta Mario Bros. -pelistä. Se julkaistiin samaan aikaan Wii U:n kanssa. Pelissä on mahdollista pelata jopa viidellä pelaajalla yhtäaikaisesti, joista neljä voi pelata tavallisella ohjaimella. Yksi voi käyttää GamePadia. Peli sisältää myös suoritettavia haasteita. (Moby Games 2014a.)

Peliin on myös itsenäinen lisäosa nimeltä New Super Luigi U. Se julkaistiin kesällä 2013. Marion sijaan päähahmona on pelin mukaisesti Luigi, joka voi hypätä korkeammalle ja pysähtyy hitaammin. New Super Luigi U sisältää uudelleen suunnitellut tasot, joiden vaikeusaste on korkeampi. (Moby Games 2014b.)

Moninpeli toimii siten, että jokainen pelaaja voi valita itselleen yhden neljästä hahmosta. Hahmot näkyvät samalla ruudulla, ja jokainen pelaajista liikuttaa omaa hahmoaan. (WittzGaming 2012.) GamePadia käyttävä pelaaja voi asettaa näytölle palikoita, joiden päälle muut pelaajat voivat hypätä (kuva 6). Kosketusnäytöllä voi myös jossain määrin vuorovaikuttaa ympäristön kanssa (TheBitBlock 2013a). Sinällään avustavan pelaajan rooli on erikoinen, koska hän ei hallitse pelissä hahmoa. GamePadissa on tarvittavat näppäimet hahmon hallitsemiselle, joten olisi ollut mahdollista, että yksi pelaajista olisi voinut olla mukana pelissä ja avustaa yhtä aikaa. Rooli jää ikävä kyllä kovin passiiviseksi, eikä sillä ole pelissä suurta merkitystä.

Kuva 6. Peliohjainten käyttäminen New Super Mario Bros. U -pelissä (Groenendijk 2012).

4.2 Rayman Legends

Rayman legends on jatko-osa Rayman Origins -pelille, ja se julkaistiin usealle alustalle vuonna 2013. Se on kaksiulotteinen tasohyppelypeli. Pelissä pelastetaan Teensie-olioita ja edetään taso kerrallaan. Hahmoilla on käytössään erityistaitoja pelin alusta alkaen, esimerkiksi hyökkäyksiä ja liikkeitä. Tasoja on erilaisia: pitkissä tasoissa täytyy löytää kymmenen Teensie-oliota. Osa tasoista on nopeatempoisempia ja lyhyempiä, esimerkiksi musiikin inspiroimia kenttiä tai taisteluita isoja vihollisia vastaan. Pääpaino onkin klassisessa tasohyppelypelin haasteissa: tasojen etenemisessä, esteiden ja vihollisten päihittämisessä sekä tutkimisessa. Tavallisissa tasoissa on mahdollista pelata Marion tyyliin, jolloin monta pelaajaa on yhtäaikaisesti pelissä tasohyppelyn roolissa. Peli tarjoaa myös Kung Foot -minipelin, jossa kaksi joukkuetta potkii ja lyö palloa ja toisiaan saadakseen pisteitä. (Nintendo Life 2013.)

Peli sisältää kaksitoista tasoa, joissa pelataan Murfy-hahmolla. Se on pieni lentävä olio, jota hallitaan Wii U:lla GamePadin kosketusnäytön avulla (kuva 7). Moninpelissä toisen on mahdollista hallita tätä hahmoa, jolloin tavallinen pelaa-

ja hoitaa tasohyppelyn. Yksinpelissä tasohyppelyä ohjaa tekoäly. Murfy voi muun muassa leikata naruja, ampua katapulteilla, kutittaa vihollisia sekä liikuttaa ja kääntää tavaroita kentällä. (Nintendo Life 2013.)

Kuva 7. Peliohjainten käyttäminen Rayman Legends -pelissä (Nintendo Master 2014).

Rayman Legends on ilmeisesti panostanut siihen, missä Mario ei onnistunut. Pelikuvan perusteella (Abrego 2013) co-op-pelaaminen on havaintojeni mukaan huomattavasti monipuolisempaa näissä tasoissa, sillä GamePadia käyttävä pelaaja voi vaikuttaa huomattavasti enemmän ympäristöön. Tämän pelaajan rooli on välttämätön etenemisen kannalta – toisin kuin New Mario Bros. U -pelissä. Näiden tasojen lisääminen on moninpelin kannalta omasta mielestäni erittäin hyvä idea, koska niiden läpäiseminen vaatii kommunikaatiota ja yhteistyötä.

Rayman Legendsistä (ja New Super Mario Bros. U:sta) voi oppia, että asymmetrisessä pelissä toisen pelaajan olisi hyvä kokea roolinsa mielekkääksi. Jos rooli jää pieneksi, toisella pelaajalla ei ole paljon merkitystä. Tämä on hyvä ottaa huomioon suunnitellessa asymmetrisiä rooleja. Rayman Legends onnistuu mielestäni asymmetrisyydessä erityisen hyvin, sillä erillisten tasojen läpäiseminen vaatii GamePadin käyttöä toiselta pelaajalta. GamePadia käyttävä pelaaja ei myöskään jää passiiviseksi katselijaksi tavallisissa tasoissa, vaan tasohyppe-

ly jatkuu normaalisti. Tekemistä riittää molemmille pelaajille.

4.3 Game & Wario

Game & Wario (2013) on kokoelma minipelejä, joita pelataan GamePadilla. Pelissä on 16 minipeliä, joista kymmenen on yksinpelejä. GamePadin käyttötapa vaihtelee pelistä riippuen. Sitä käytetään moninpeleissä vuorotelleen. Minipelien hahmot ovat Wario-peleistä. (Moby Games 2014c.)

4.3.1 Taxi, KungFu ja Fruit (eri näkymät, tarkkailu)

Taxi-minipelissä toinen pelaajista ohjaa taksia GamePadilla ensimmäisen persoonan näkökulmasta. Pelaaja voi ampua tykillä avaruusaluksia ja muita kohteita. Kartalla on muun muassa arkkuja, joita pitää kerätä. Televisioruudulla on näkymä pelistä ylhäältä päin kuvattuna (kuva 8). Toinen pelaajista toimii neuvonantajana ja voi kertoa, missä viholliset ja aarteet sijaitsevat. Pisteitä pelissä saadaan kerättyjen tavaroiden ja ajan perusteella. (TheBitBlock 2013b.) Tämä rooli on siinä mielessä mielenkiintoinen, että toinen pelaajista ei varsinaisesti vuorovaikuta pelin kanssa millään tavalla. Hän vain ohjeistaa GamePadilla pelaavaa. Periaatteessa tämän pelin pelaaminen onnistuisi yksinkin, sillä mikään ei varsinaisesti estä välillä televisioruudulle katsomista. Kahdestaan työskentely on kuitenkin tehokkaampaa – silloin voi keskittyä yhteen asiaan kerrallaan.

KungFussa käytetään jälleen pelien välillä erilaista näkymää, perusperiaate on samantapainen kuin Taxissa. Televisioruudulla on kuitenkin tällä kertaa kuva pelistä perspektiivissä. GamePadilla pelaava näkee pelin ylhäältä kuvattuna, mutta alue on hyvin rajattu. Hän ohjaa hahmoa kallistelemalla laitetta. Hän ei siis näe kovin hyvin kauas, eikä tiedä missä kerättävät tavarat ovat. Toinen pelaajista on taas neuvonantajan roolissa. Hän auttaa GamePadia käyttävää pelaajaa löytämään tavarat. (TheBitBlock 2013b.)

Toisen pelaajan rooli on KungFussa merkittävä, sillä tasolta toiselle hyppiminen vaatii paljon keskittymiskykyä ylhäältä kuvattuna. Toisaalta toinen pelaajista ei ole pelaavassa roolissa. Taxissa ja KungFussa on samantapainen idea erilaisesta näkökulmasta peliin, jolloin toinen pelaaja voi antaa ohjeita. Sinällään harmilliseksi mielestäni osoittautuu se, että neuvonantaja ei ole itse pelimaailman kanssa suorassa vuorovaikutuksessa. Toisaalta kyse on vain minipelistä, joten se lienee hyväksyttävää lyhyemmän peliajan vuoksi.

Kuva 8. Eri näkymät Taxi-pelissä (Nintendo 2014e).

Fruit on tarkkaavaisuutta ja oveluutta vaativa peli. GamePadilla pelaava valitsee ihmishahmon neljäntoista vaihtoehdon joukosta. Jokainen hahmoista näyttää erilaiselta. Hän ohjaa hahmoa pelissä vilkkaan väkijoukon seassa ja koettaa kerätä hedelmiä. Peliruudulla on näkymä pelistä, mutta toisen pelaajan tarkoitus on etsiä GamePadia käyttävän pelaajan hahmo väkijoukosta. Pelissä on aikaraja, mutta toinen pelaaja voi arvata tai valita hahmon missä vaiheessa tahansa näiden neljäntoista henkilön joukosta. (TheBitBlock 2013c.) Käytännössä toisella pelaajalla on siis tarkkailijan rooli, joka on aktiivinen ja merkittävä pelin kan-

nalta. Asymmetrisyys on selkeää pelaajien välillä.

4.3.2 Arrow, Patchwork, Islands & Ski (GamePadin käyttö ohjaimena)

Arrow-minipelissä televisioruudulla on kenttä, johon ilmestyy vihollisia. Ne täytyy ampua jousella, jolloin saa pisteitä. GamePadia käytetään vuorotellen. Sitä pidetään vaakatasossa ruutua kohti, ja näytöllä on jousi (kuva 9). Kun kosketusnäyttöä pyyhkäisee alaspäin, jousi virittyy. Kun kosketuksen päästää irti, nuoli ammutaan. Suunta määräytyy GamePadin kallistuskulman perusteella. Jos viholliset pääsevät ruudun alalaitaan asti, ne hyppäävät GamePadin ruudulle ja koettavat viedä pelaajan mansikat, jotka toimivat elinvoimana. Viholliset voi nitistää koskettamalla niitä näytöllä. GamePadia voi myös käyttää kilpenä kääntämällä sen pystyasentoon, jolloin ammukset kimpoavat siitä takaisin. Myös Patchwork-pelissä hyödynnetään kosketusnäyttöä. Stylus-kynällä voidaan liikutella palapelin palasia ja asettaa ne näytölle kuvioiden muodostamiseksi. (TheBitBlock 2013d.)

Kuva 9. GamePadin käyttäminen Arrow-pelissä (Davidson 2013a).

Arrowissa on hyödynnetty GamePadia erikoisella tavalla, kosketusnäyttö toimii kätevänä ohjaimena. Peli hyödyntää GamePadin kallistuskulmaa sekä kosketusnäyttöä. Sinällään minipelissä ei kuitenkaan hyödynnetä asymmetrisiä rooleja, sillä sitä pelataan vuoro kerrallaan. Asymmetrisyyden toteuttamiseksi toinen pelaajista voisi käyttää jotakin toista asetta tai tuhota tiettyjä kohteita.

Islandsissa GamePadia käytetään Arrowin tapaan ampumiseen. Tässä tapauksessa se toimii linkona. Tarkoitus on ampua palikan näköisiä olentoja puolipallon muotoiselle saarelle, jonka asento vaihtelee sille asetetun painon perusteella. Palikat voivat pudottaa toisiaan saarelta. Ne voivat pudota myös saaren kallistuessa liikaa toiselle puolelle. Saaren sektorit määrittelevät, kuinka paljon niillä pysyvistä olennoista saa pisteitä. Kumpikin pelaajista ampuu GamePadilla vuorotellen. Voittaja on se, jolla on pelin lopussa eniten pisteitä. (TheBitBlock 2013e.) Pelaajien roolit eivät ole asymmetriset, mutta laitteen käyttö on mielestäni kekseliästä.

Ski-minipelissä GamePadia käyttävä ohjaa laskettelevaa hahmoa ylhäältä kuvattuna (kuva 10). Ohjaus tapahtuu laitetta kallistamalla vasemmalle ja oikealle. Jälleen kerran GamePadia käytetään ohjaimena. Televisiossa näkymä on kolmannen persoonan perspektiivissä. Tarkoitus on kerätä mahdollisimman monta naislaskettelijaa mukaansa reitin varrelta. (TheBitBlock 2013c.) Vaikuttaa siltä, että toisen pelaajan rooli jää pieneksi, koska GamePadilta näkyy kerättävien kohteiden sijainnit. Näkymä on tosin jossain määrin rajattu. Pelaaminen voisi onnistua kallistelemalla GamePadia ja katsomalla suurelta näytöltä yhtä aikaa.

Kuva 10. Pelaajan hahmon hallinta GamePadilla Ski-pelissä (Davidson 2013b).

4.3.3 Gamer (erilliset minipelit)

Gamer on varsin erikoinen minipeli. Televisioruudulla on näkymä huoneesta, jossa poika pelaa yöllä käsikonsolia. GamePadia käyttävä pelaa mikropelejä (minipelejäkin pienempiä pelejä), hän on tavallaan tämän pojan roolissa. Tarkoitus on pelata pelejä äidiltä salassa. Hän käy tarkastamassa huoneen silloin tällöin. Myös muita häiriöitä voi ilmaantua, esimerkiksi kissa. Pojan täytyy lopettaa pelien pelaaminen, ettei hän jää kiinni. (TheBitBlock 2013e.) Toisella pelaajalla on siis tarkkailijan rooli, hänen täytyy varoittaa mikropelejä pelaavaa. Asymmetrisyys on tässä pelissä erittäin selkeää. GamePadin näkymä on täysin erilainen, siinä on kokonaan toisia pelejä.

4.4 Nintendo Land

Nintendo Land on toinen kokoelma minipelejä Wii U -konsolille. Premium-paketin ostajat saavat pelin konsolin mukana. Pelin teema on virtuaalinen huvipuisto, jossa on kaksitoista minipeliä. Puolet peleistä on yksinpelejä, loput moninpelejä. Minipelien teemat tulevat Nintendon julkaisemista peleistä, esimerkiksi Mariosta, Zeldasta ja Metroidista. Monet peleistä hyödyntävät GamePadin toiminnallisuuksia. (Moby Games 2014d.) Alla kuvatut osiot perustuvat näihin minipeleihin.

4.4.1 Kilpailu ja vastakkainasettelu asymmetrisissä peleissä

Luigi's Ghost Mansion tarjoaa asymmetrisen pelikokemuksen. GamePadia käyttävä pelaaja hallitsee kummitusta pimeässä huoneessa (kuva 11). Se on näkymätön television ruudulla. Muilla pelaajilla on taskulamppu, jonka avulla he yrittävät löytää ja tuhota kummituksen. GamePadilla pelaavan rooli on kummituksena ottaa kiinni kaikki muut pelaajat. (Moby Games 2014d.) Pelistä erityisen mielenkiintoisen tekee sen vastakkainasettelu – roolit on jaettu mielenkiintoisella tavalla. Muiden pelaajien tulisi tehdä yhteistyötä, että kummitus löytyy. Toisaalta taas GamePadia käyttävä pelaaja voi yrittää toimia ovelasti.

Kuva 11. Pelinäkymä Luigi's Ghost Mansion -pelistä (Edge 2013).

Animal Crossing Sweet Day hyödyntää samaa mallia. Pelaajat yrittävät voittaakseen kerätä mahdollisimman paljon makeisia eläinhahmoillaan. GamePadia käyttävä pelaaja hallitsee kahta vartijaa. Kumpaakin niistä voi hallita erikseen analogisilla tateilla. Vartijat voivat lyödä pelaajia, jolloin karkkeja putoaa maahan. (Moby Games 2014d.) Tavallisilla pelaajilla vaikuttaa olevan huomattava etu vartijoihin nähden, sillä kahden hahmon hallitseminen yhtä aikaa voi olla vaikeampaa. Rooli on ilmeisesti rajoitettu yhteen pelaajaan, mikä on harmillista. Olisi varmaan ollut mielekkäämpää, että kaksi pelaajaa toimii vartijoina.

4.4.2 Pelaajien eriarvoisuus

Zelda Battle Quest -pelissä valitaan rooli miekkamiehen tai jousiampujan välillä. Vain GamePadilla voi ampua nuolia (kuva 12). Pelaajien täytyy tehdä yhteistyötä päästäkseen etenemään, koska elinvoima on jaettu pelaajien kesken. (Moby Games 2014d.) Pelissä on tarkoitus päihittää kaikki viholliset, suorittaa tehtäviä ja löytää Zelda-maailmasta tuttu Triforce (BNGR 2012).

Kuva 12. Peliohjainten käyttäminen Zelda Battle Quest -pelissä (Plant 2012).

Pikmin Adventure on monen pelaajan brawler-peli. Lajityypin mukaisesti siinä ollaan lähitaistelussa vihollisia vastaan. GamePadia käyttävä saa hahmokseen Captain Olimarin, muut pelaajat ovat Pikminejä. Pelissä voi kerätä nektaria, jonka avulla hahmot saavat uusia tasoja. Elinvoima on jaettu samaan tapaan kuin Zelda Battle Questissa. (Moby Games 2014d.)

Metroid Blastissa yksi pelaajista voi lentää GamePadia käyttäen avaruusaluksella, jossa on tykit. Alusta ohjataan gyroskoopin avulla. Muut pelaajat käyttävät tavallisia hahmoja maan pinnalla. Pelissä on erilaisia tehtäviä, ja se tarjoaa kilpailuhenkisiä ja yhteistoiminnallisia pelimoodeja. (BNGR 2012). Yhteistyö näyttääkin olevan erittäin keskeinen osa asymmetrisiä pelejä. Edellisistä esimerkeistä käy ilmi, että se on hyvä tapa toteuttaa yhteistoiminnallisia pelejä.

4.5 Muita pelejä

Mario Kart 8 on perinteinen peli, jossa ajetaan Nintendon pelihahmoilla. Tarkoitus on luonnollisesti päästä ensimmäisenä maaliin. Pelissä on 32 rataa. (Moby Games 2014e.) GamePadia käytetään pelissä hyödyksi kolmella tavalla. Ensinnäkin, useimpien Wii U -pelien tapaan Mario Kartia voi pelata GamePadin ruudulta. Lisäksi kiihtyvyysanturin avulla voidaan ohjainta käyttää rattina: ajoneuvo kääntyy vasemmalle tai oikealle riippuen suunnasta. Kolmas käyttötarkoitus on näyttää kartta kentästä sekä pelaajan sijoitus kisassa. (IGN 2014b). Pelin ei siis välttämättä tarvitse olla asymmetrinen hyötyäkseen mobiililaitteen ominaisuuksista.

The Legend of Zelda: The Wind Waker julkaistiin alun perin GameCubelle. Se sai kuitenkin uusintaversion Wii U -konsolille, jonka myötä peli sai tuen GamePadille. The Wind Waker on samaa luokkaa muiden The Legend of Zelda -pelien kanssa. Se sisältää toimintaa taisteluiden muodossa, ongelmanratkaisua ja kevyitä roolipelielementtejä. (Moby Games 2014f.) GamePadin ruudulla on näkymä tavaroille, kartoille ja pulloille. Lisäksi kamera käyttää hyödykseen joissakin tapauksissa GamePadin kiihtyvyysanturia. Esimerkiksi teleskoopilla voi katsella ympäriinsä ja tarkentaa kääntämällä ohjainta. (Children's Technology Review 2013.) Mielestäni tämä on hyvä esimerkki mobiililaitteen hyötykäytöstä yk-

sinpelissä. Tosin Wii U:n etu tablet-tietokoneeseen verrattuna on fyysiset näppäimet – jotkin toiminnallisuudet voivat olla hankalia usealla ohjauslaitteella kerrallaan.

4.6 Yhteenveto GamePadin käytöstä (mobiililaitteen hyödyntäminen)

Kuten esimerkeistä kävi ilmi, GamePadia tai mobiililaitetta voidaan hyödyntää peleissä monella tavalla. Esineitä voidaan asettaa peliin kosketusnäytön avulla ja erilaisia tietoja pelistä voidaan näyttää ruudulla. Mobiililaitetta voidaan käyttää myös ohjaimena tai toisena näyttönä, jossa voidaan näyttää erilainen näkymä pelistä (eri kuvakulma tai jopa eri ulottuvuus). Käyttötapaukset koskevat myös älypuhelinta tai tablettia, joten niitä voidaan hyödyntää myös PC:llä. Monesti toisen laitteen hyödyntäminen vaikuttaa myös pelaajien välisiin rooleihin.

1) Esineiden asettaminen peliin kosketusnäytön avulla

Kosketusnäytöllä on mahdollista luoda peliin objekteja, esimerkiksi esteitä tai tavaroita, joilla voi tuhota vihollisia. Tämä antaa toiselle pelaajalle merkittävästi erilaisen roolin. Huomattavaa on kuitenkin, että vähäisen toiminnan vuoksi tämä ei välttämättä ole tarpeeksi mielenkiintoista. Olisi tärkeää, että myös avustava pelaaja kokee itsensä hyödylliseksi.

2) Tiedon näyttäminen ruudulla

Toista näyttöä voidaan käyttää ylimääräisen tiedon näyttämiseen. Esimerkiksi kartta, pelin tavarat tai tilanne pystyvät olemaan jatkuvasti näkyvillä. Tietojen näyttämistä hyödynnettiin jo GBA-käsikonsolin avulla. On suuri etu, että tärkeät tiedot voivat olla jatkuvasti saatavilla. Tavallisesti pelaajan täytyy avata erikseen käyttöliittymä esimerkiksi katsoakseen karttaa tai hallitakseen tavaraluetteloa. Lisäksi kosketusnäytöllä on mahdollisesti intuitiivisempi käsitellä erilaisia valikoita. Sinällään se ei tuo paljon uutta, sillä valikot ja tiedot voidaan toteuttaa tavalliseen käyttöliittymään peliruudulle. Yhtäaikaisuudesta voi kuitenkin olla joissakin tapauksissa hyötyä.

3) Laitteen käyttäminen ohjaimena

Mobiililaite avaa paljon mahdollisuuksia pelin ohjausta varten. Kosketusnäyttöä voidaan käyttää monella tapaa. Näyttöä pyyhkäisemällä voidaan esimerkiksi ampua esineitä jousella tai lingolla pelinäkymään. Kosketusta voidaan käyttää esineiden asettamiseen tai liikuttamiseen ruudulla. Kiihtyvyysantureiden avulla voidaan havaita laitteen kääntyminen. Mobiililaitteen asentoa voidaan käyttää siis hyväksi. Esimerkiksi laitetta kääntämällä voidaan tähdätä tai vaihtaa pelaajan hahmon liikkumissuuntaa.

Mikrofoni ja kamera ovat myös käytettävissä. Kameran avulla on mahdollista lisätä vuorovaikutteisuutta tosielämän kanssa. Mikrofonin avulla pelaaja voi esimerkiksi ohjata peliä äänen avulla tai tallentaa äänitteitä.

Huomattava ero GamePadin ja mobiililaitteen välillä on se, että kosketusnäytöllisissä puhelimissa ei ole fyysisiä pelinäppäimiä tai analogisia tatteja. Tämä asettaa omat haasteensa mobiililaitteen käyttämiseen ohjaimena. Moninpelissä toisella pelaajalla on koko laite käytössään, mutta yksin kosketusnäyttöä voi olla vaikea käyttää yhtäaikaisesti tietokoneen näppäimistön ja hiiren tai peliohjaimen kanssa.

4) Eri näkymät

Näkökulma laitteiden välillä voi olla erilainen. Ensimmäisen tai kolmannen persoonan perspektiiviä käytetään yleisesti peleissä. Jos toisen laitteen näkymä on erilainen verrattuna itse peliin, voidaan rooleista tehdä asymmetriset. Toisella pelaajalla on esimerkiksi ylänäkymä (joko pienellä tai isolla ruudulla), joka antaa edun. Tämän avulla voidaan tehdä esimerkiksi kommunikaatiota vaativia pelejä. Toinen pelaajista kertoo, missä tavarat sijaitsevat tai minne täytyy mennä.

Näytöllä voi myös olla kokonaan eri näkymä pelistä, esimerkiksi toisesta paikasta tai ”ulottuvuudesta”. Se, mitä toinen pelaaja tekee tässä näkymässä vaikuttaa ensimmäisen pelaajan peliin. Esimerkiksi mobiilinäytölle voisi ilmestyä vihollisia, jotka täytyy tuhota. Ne ovat myös toisen pelaajan pelissä, mutta ne voi tuhota

pelkästään kosketusnäytön avulla.

5) Toinen näyttö

Perinteisesti yhteistoiminnallisia pelejä pelataan joko jaetulla näytöllä tai useammalla laitteella. Sen sijaan toinen pelaajista voi käyttää ylimääräistä näyttöä pelatessa. Jotkut pelit eivät käytä jaettua ruutua, mutta silloin ongelmaksi voi muodostua toisen pelaajan katoaminen ruudulta tai rajojen asettaminen pelinäkymälle. Jaetulla näytöllä taas pelistä näkyy pienempi osa, ja kumpikin pelaaja pystyy näkemään toisensa (vaikka näin ei olisi tarkoitus). Toista näyttöä käyttämällä näiltä ongelmilta voidaan välttyä.

6) Minipelit

Kohtaa 4 sivuten (eri näkymät), toisen laitteen avulla on mahdollista pelata kokonaan irrallisia tai pelkästään osittain itse pääpeliin liittyviä minipelejä. Ne voivat olla esimerkiksi pelistä löytyviä salaisuuksia. Minipeleistä voisi saada esimerkiksi tavaroita tai muita palkintoja.

Hyvä esimerkki irrallaan olevasta minipelistä on Final Fantasy 8 -pelin Chocobo World. PC-versiossa erillisessä ohjelmassa tai Japanissa Pocket Stationilla oli mahdollista pelata peliä, jossa hallitaan Chicoboa (pientä pelimaailman lintua). Se voi liikkua kartalla ja löytää matkallaan tavaroita sekä taistella vihollisia vastaan. Tavaroita pystyi siirtämään itse pääpeliin. (Final Fantasy Wiki 2014.)

5 Asymmetrisyys ja mobiililaitteiden hyödyntäminen PC:llä

Tällä hetkellä PC:llä asymmetrisyydessä ei keskitytä niinkään käyttämään ylimääräisiä laitteita, vaan moninpeleissä keskitytään enemmän itse asymmetrisiin rooleihin. Mobiililaitteita hyödynnetään myös muutamalla muulla tavalla, ja ne esitellään tässä osiossa.

5.1 PC:n ja mobiililaitteen välinen suhde

OnLive tarjoaa mahdollisuuden käyttää (virtuaalista) Windows-työpöytää mobiililaitteella. Se tarjoaa Microsoft Wordin, Excelin ja Adobe Readerin sekä 2 gigatavua tallennustilaa ilmaiseksi. (OnLive 2014a.) Lisäksi OnLive tarjoaa pilvipalvelua pelien pelaamiseksi mobiililaitteilla. Palvelu toimii siten, että OnLiven palvelimet suorittavat pelit paikallisesti ja lähettävät kuvan pelistä mobiililaitteeseen Internetin välityksellä. Mobiililaitteeseen asennetaan asiakasohjelma. (OnLive 2014b.) Tämä mahdollistaa laskennallisesti vaativien pelien pelaamisen mobiililaitteessa (kuva 13). Pelien graafinen laatu on huomattavasti parempi PC- kuin mobiilipeleissä. Yritys tarjoaa myös ohjaimia, joita voidaan käyttää televisiolla (OnLive 2014c).

Tavallaan palvelu muistuttaa PS4-konsolin Remote Play -toimintoa ja Wii U:n GamePadia, joita voidaan käyttää pelien pelaamiseksi mobiililaitteella muualla kuin konsolin ääressä. Suuri ero on pelien suorittamisessa yrityksen omilla palvelimilla, jolloin käyttäjä ei tarvitse muuta kuin mobiililaitteen tai TV-järjestelmän pelatakseen pelejä. Konsepti on mielenkiintoinen, joskin käyttäjä on riippuvainen yrityksen tarjonnasta. Lisäksi PC-pelejä voi olla hankala pelata mobiililaitteilla, koska niitä ei ole suunniteltu niillä pelattavaksi. Yritys tarjoaakin ongelmaan peliohjaimia, jotka toimivat Android-laitteissa (OnLive 2014c). Palvelun kautta on mahdollista pelata jo omistamiaan pelejä tai valita yli 250 pelin valikoimasta (OnLive 2014d). Palvelu tekee yhteistyötä digitaalisia pelejä myyvän Steamin kanssa (ExtremeTech 2014). Myös Kainy-sovelluksen avulla on jopa mahdollista pelata PC-pelejä mobiililaitteella tai käyttää sitä ohjaimena lähiverkon tai Internetin välityksellä (Kainy 2014).

Kuva 13. PC-peli tablet-tietokoneella OnLive-palvelun välityksellä (OnLive 2014e).

Power-Grid on Roccatin yritys hyödyntää älypuhelimia PC:llä ohjelmiston avulla. Se on ilmainen sovellus, jolla on mahdollista hallita tietokonetta älypuhelimen avulla. Power-Grid on eräänlainen muokattava kaukosäädin tietokoneelle. Puhelimeen ladataan ohjelmisto, jonka kautta laitteet voivat kommunikoida keskenään. Ohjelmaa voidaan käyttää peleissä, ja sitä voidaan käyttää myös sosiaaliseen kommunikaatioon. (Roccat 2014a.) Roccat tarjoaa pelinkehittäjille käyttöön Power-Grid ohjelmistokehityspaketin (Roccat 2014b).

5.2 PC-pelit

Jotkut PC-pelit (erityisesti suuret tuotannot) ovat hyödyntäneet mobiililaitteen ominaisuuksia, etenkin mahdollisuutta käyttää sitä toisena näyttönä. Tämä tapahtuu erikseen ladattavan sovelluksen avulla (companion app).

Watch Dogs -pelissä mobiililaitteilla voidaan pelata kaikkien alustojen välillä. Kartalta voi nähdä missä muut pelaajat ovat ja kutsua heidät pelaamaan. Haasteessa ensimmäinen pelaaja ajaa autolla pelinäkymässä mahdollisimman nopeasti pisteeltä toiselle, kun toinen hallitsee mobiililaitteella poliisivoimia ja yrit-

tää estää tämän pelaajan liikkumisen ylänäkymästä (kuva 14). (GT News 2014.) Kamppailussa on pelaajien välillä selvät asymmetriset roolit.

Kuva 14. Laitteiden näkymät Watch Dogs -pelissä (Splechta 2014).

Grand Theft Auto V:ssä voidaan käyttää iFruit-sovellusta mobiililaitteella. Se sisältää eräänlaisia alisovelluksia, joiden avulla on mahdollista vuorovaikuttaa pelin kanssa. Los Santos Customs mahdollistaa auton muokkaamisen. Sillä voidaan vaihtaa esimerkiksi moottorin kokoa, vanteita, väriä ja rekisterikilven ulkonäköä. Pelissä olevaa koiraa (Chop) voidaan hoitaa ja kehittää iFruitin kautta. Tämä tapahtuu minipelejä pelaamalla. Sovelluksesta voidaan valita myös koiralle kauluspanta. Kultaisen kauluspannan saa voittamalla kultaisen mitalin jokaisesta minipelistä. Tämän lisäksi iFruitissa voi tarkastella pelin tapahtumia, radioasemia, kuvia pelistä sekä henkilöiden profiileja. Sovelluksessa on myös osiot pelien ja oheistuotteiden ostamiseksi. (IGN 2014c.)

PlanetSide 2 -pelissä mobiilisovellusta käytetään pelioppaana. Se sisältää tietoa esimerkiksi aseista, sotilasluokista, ajoneuvoista ja saavutuksista. Sovelluksen avulla voi tutkia hahmojen profiileja sekä tarkkoja pelisijoituksia aihealueen perusteella. Suurin ominaisuus on interaktiivinen kartta, jossa on reaaliaikaista tietoa pelialueista, resursseista ja rakennuksista. Lisäksi sovellusta voi käyttää äänellä puhumiseen. Sovellusta voidaan käyttää siis pelaamisen ohessa. (Appburst 2014.)

World of Warcraftin Armory on pelin pelaajille hyödyllinen sovellus. Sen avulla voi tarkastella massiivisen nettiroolipelin hahmojen varustuksia ja tietoja. Pelin huutokauppaa voidaan käyttää mobiililaitteella sovelluksen avulla, mikä on valtava etu. Pelaajan ei tarvitse kirjautua tietokoneella peliin päästäkseen hallitsemaan tavaroiden ostamista ja myyntiä. Sovellus sisältää kiltaan liittyviä toimintoja, muun muassa tietoja killasta, keskustelun ja yksittäisten viestin lähettämisen jäsenten kesken. Lisäksi sovelluksessa on kalenteri pelin tapahtumille, killan uutisille ja saavutuksille. (Blizzard Entertainment 2014.)

Frozen Synapse on vuoropohjainen strategiapeli. Tarkoitus pelissä on hallita joukkoa sotilaita kentällä, jossa on muun muassa käytäviä, huoneita sekä suojia. Jokaisella sotilaalla on oma ampuma-ase, jolla on tietyt ominaisuudet. Moninpeliä on mahdollista pelata PC:n ja mobiililaitteen välillä. Tarkoitus on yrittää tuhota vastapuolen joukot. (Moby Games 2014g.) Pelitila tallennetaan palvelimelle, jolloin esimerkiksi PC:llä aloitettua peliä on mahdollista jatkaa tablet-tietokoneella. Tosin silloin täytyy omistaa sekä pelin PC- että mobiiliversio. Pelaaminen onnistuu kuitenkin samalla pelitunnuksella. (Mode 7 2013.)

Tom Clancy's The Division on yksi harvoista PC-peleistä, joka pyrkii hyödyntämään asymmetrisyyttä merkittävällä tavalla. Mobiilipelajat voivat liittyä mukaan taisteluihin, jolloin tarkoitus on olla avustavan robotin roolissa. Silloin on mahdollista parantaa haavoittuneita joukkuetovereita sekä auttaa taistelussa rakettiiskulla. Robotilla on erilaisia taitoja, se pystyy heikentämään suojia ja nostamaan joukkueen aiheuttamaa vahinkoa. Mobiililaitteella pelataan ylhäältä päin kuvattuna, jolloin pelaaja näkee pelin tilanteen eri näkökulmasta (kuva 15). (TheSixthAxis 2013.)

Kuva 15. Mobiilinäkymä Tom Clancy's Division -pelistä (Sage 2013).

Sinänsä asymmetrisyys on PC:llä aika vähäisiä, vaikka mobiililaitteita on hyödynnetty apusovellusten myötä. Asymmetristen roolien toteuttamiseksi ei kuitenkaan välttämättä tarvita mobiililaitteita, vaikka esimerkiksi Wii U pyrkii hyödyntämään pelisuunnittelussa GamePadia. PC:lle on kehitetty joitakin pelejä, jotka hyödyntävät itse konseptia muulla tapaa.

Evolve on hyvä esimerkki tällaisesta pelistä. Pääidea Hunt-pelitilassa on taistella neljän hengen joukkueena yhtä todella suurta hirviötä vastaan räiskintäpelien tyyliin. Jokaisella joukkueen pelaajista on tietty rooli: parantaja, tuhoaja, avustaja tai loukuttaja. Kaikkia rooleja tarvitaan hirviön päihittämiseksi. Kentällä on myös muita vihollisia, joita tappamalla hahmoja voi kehittää. Yksi pelaa koko joukkuetta vastaan hirviöllä, joka on todella voimakas vastustaja. Erityisesti pelin alussa on tarkoitus kehittää hirviö mahdollisimman voimakkaaksi metsästämällä muita hirviöitä. Voimatasoja on kaiken kaikkiaan kolme. Mitä voimakkaammaksi hirviö ehtii kasvaa, sitä vaikeampi muiden pelaajien on päihittää se. (Kotaku 2014.)

Natural Selection 2: Combat -ammuskelupelissä pelaaja voi valita, haluaako pelata ihmisillä (Marines) vai muukalaisrodulla (Kharaa). Kumpikin vastapelaaja valitsee rodun, joilla on omalaatuiset ominaisuudet sekä kyvyt. Tämä tekee pelistä asymmetrisen, koska rotuja pelataan eri tavalla. Ihmiset käyttävät raskaita aseita ja suojia. Muukalaiset taas erikoistuvat hiipimiseen, nopeuteen ja voimaan. Ne voivat kehittyä pienestä oliosta viideksi eri muodoksi. Jokaisella evoluutiolla on omat kykynsä. Pelaajat saavat pelatessaan kokemuspisteitä, joita voidaan käyttää hahmon kehittämiseen. (Faultline Games 2014.)

5.3 Laitteistoratkaisut

PC:lle on kehitelty joitakin laitteistoratkaisuja, joilla pyritään integroimaan mobiililaitteiden käyttö pelien kanssa. Lisäksi PC:lle on saatavilla joitakin lisälaitteita älypuhelinten ja tablettien lisäksi, joita voidaan käyttää asymmetrisen pelikokemuksen luomiseksi.

Yksi varhaisista yrityksistä hyödyntää laitteistoa tietojen näyttämiseksi PC:llä on Logitechin GamePanel, joka on tietyissä näppäimistöissä oleva pieni näyttö (kuva 16). Sen avulla voidaan näyttää peli- ja järjestelmätietoja. Näyttöä varten on tehty ohjelmistosovelluksia. (Logitech 2014.) GamePanel on G19-mallissa (2009) käännettävä ja siinä on 320 x 240 pikselin resoluutio. Se tuki julkaisuaikana yli kuuttakymmentä peliä, muun muassa World of Warcraftia. (TechPowerUp 2009.) Vuonna 2014 Logitech julkisti minimalistisen K480-näppäimistön, johon älypuhelimen voi laittaa telakkaan. Sen avulla voi kirjoittaa PC:lle, puhelimeen tai tablettiin bluetooth-yhteyden avulla. (Honorof 2014.) Muuta erityistä toiminnallisuutta sillä ei tosin ole.

Kuva 16. GamePanel-nestekidenäyttö Logitech G19 -näppäimistössä (AnandTech 2013).

Roccat on pyrkinyt yhdistämään älypuhelimia ja tietokoneita. Tulevassa Skeltr-näppäimistössä on älypuhelinta varten teline, joka sijaitsee näppäimistön takareunassa (kuva 17). Telinettä pystyy liikuttamaan ja kääntämään. Laite on yhteensopiva Bluetoothin kanssa. Laitteen avulla on mahdollista kommunikoida PC:n kanssa ja käyttää älypuhelimen toimintoja. (TechPowerUp 2014.) Tämä luo uuden mahdollisuuden hyödyntää älypuhelimia PC-pelaamisessa. Näytöltä pystyy esimerkiksi käynnistämään pelejä ja ohjelmia sekä tarkastelemaan pelien tietoja (Roccat 2014c). Verrattuna Logitechin ratkaisuun älypuhelimen käyttö tuntuu luonnollisemmalta niiden yleisyyden vuoksi – erillistä näyttöä ei siis tarvita laitteistossa.

Nvidia Shield on mielenkiintoinen ratkaisu PC-pelien pelaamiseksi mobiililaitteilla. Se on erityisesti pelikäyttöön tarkoitettu järjestelmä. The Ultimate Gaming portablessa on Android-laite ja peliohjain liitetty yhteen. (Nvidia 2014a.) Sillä voi luonnollisesti pelata Android pelejä, mutta lisäksi Nvidian GameStream teknologian avulla on mahdollista pelata PC-pelejä langattoman verkkoyhteyden välityksellä. Se onnistuu kotiverkossa tai sen ulkopuolella Internetin välityksellä. Pelin kuvan voi saada näkymään myös TV-ruudulle. (Nvidia 2014b.) Erikseen on saatavilla myös pelikäyttöön tarkoitettu SHIELD-tabletti ja peliohjain (Nvidia

2014c).

Kuva 17. Roccat Skeltr-näppäimistön teline älypuhelimelle (TechPowerUp 2014).

Oculus Rift on ollut viime aikoina yksi pinnalla olevista laitteista PC-pelaamiseen liittyen. Se on päähän asetettava virtuaalitodellisuuslaite, jonka avulla peli voidaan kokea uudella tavalla. Pelaaja on sisällä pelitilanteessa, sillä päätä kääntämällä voi katsella pelitilaa samalla tapaa kuin oikeassa elämässä. Kokemus luodaan kolmiulotteisen stereoskooppisen näkymän avulla. Kummankin silmän kohdalla on siis pienet näytöt. (Oculus VR 2014.) Oculus Riftiä voitaisiin hyödyntää asymmetrisessä pelinkehityksessä luomalla rooleja, joissa yksi pelaajista käyttää virtuaalitodellisuuslaseja ja toinen esimerkiksi mobiililaitetta.

Leap Motion Control on toinen mielenkiintoinen pieni laite PC-alustalle. Sillä on mahdollista hallita tietokonetta eleillä 3D-tilassa omia käsiä käyttämällä. Näyttöä kohti voi osoittaa sormella, kättä voi heiluttaa tai sillä voi tarttua esineisiin. Käyttötarkoituksia ovat esimerkiksi Internetin selaaminen, artikkelien lukeminen, kolmiulotteisten objektien käsittely ja pelien pelaaminen käsieleitä käyttäen. Laite on hyvin tarkka, sillä voi havaita jopa millimetrin sadasosan muutokset liikkees-

sä. Laite pystyy seuraamaan käsien liikkeitä 200 ruutua sekunnissa. Se on jossain määrin samantapainen pelikonsolien eleohjausjärjestelmiin verrattuna, tosin hallinta tapahtuu käsieleiden avulla. (Leap Motion 2014.) Leap Motion Controllia voitaisiin hyödyntää asymmetristen roolien kehittämisessä PC:lle. Edellä mainittujen laitteiden perusteella voidaan todeta, että asymmetrisyyden ei tarvitse rajoittua pelkästään mobiililaitteisiin.

6 Toteutuksen menetelmät

Asymmetrisen pelikokemuksen toteuttamiseksi täytyy synkronoida vähintään kaksi pelitilaa. Tämä tarkoittaa sitä, että pelit kommunikoivat keskenään ja lähettävät tietoa toisilleen päästäkseen samaan tilaan. Lähestymistapa asymmetrisyyden toteuttamisessa on erilainen verrattuna perinteiseen verkkopelikehitykseen. Esimerkiksi massiivisissa monen pelaajan roolipeleissä (MMORPG) jokainen pelaaja on saman pelimaailman sisällä. Asymmetrisissä peleissä taas pelimaailma voi olla kummassakin laitteessa hieman erilainen niin pelimekaniikan kuin ulkonäönkin kannalta. Tämä asettaa omat haasteensa suunnittelulle. On tarpeellista tarkastella, miten synkronointi voidaan toteuttaa laitteiden välillä ja kuinka itse pelitilat toteutetaan.

6.1 Pelipalvelin

Kommunikaatio verkossa tapahtuu tietokoneesta toiselle eli asiakkaiden ja palvelimien välillä (Unity Technologies 2014b). Palvelin on tietokone, joka vastaa tiedon tallentamisesta ja lähettämisestä asiakkaalle. Asiakas ottaa yhteyden palvelimeen tekemällä sille pyynnön. Tätä kutsutaan myös asiakas-palvelin-malliksi. (Rouse 2008.) Usein palvelin käyttää erillistä käyttöjärjestelmää, ohjelmistoa ja jopa laitteistokomponentteja tätä tarkoitusta varten, mutta se ei ole välttämätöntä. Asiakas-palvelin-malli on varsin tyypillinen tapa toteuttaa verkkopeli.

Pelikäyttöön tarkoitettu palvelin voi olla keskitetty, pelkkään tietoon ja kutsuihin vastaava tietokone, johon muut pelaajat ottavat yhteyden (Unity Technologies 2014b). Usein verkkopelaaminen onnistuu kuitenkin ilman keskitettyä palvelinta. Kuka tahansa pelaajista voi toimia palvelimena, eikä erillistä tietokonetta tätä varten tarvita. Tästä hyviä peliesimerkkejä ovat Heroes of Might and Magic III (1999) ja Diablo II (2000). Niissä on mahdollista luoda verkkopeli lähiverkkoa käyttäen tai IP-osoitteen perusteella. Diablo III:ssa (2012) verkkoyhteys pelipalvelimeen on välttämätön, eikä peliä pysty pelaamaan ilman sitä.

Pelipalvelin voi olla autoritaarinen tai ei-autoritaarinen. Autoritaarinen palvelin hallitsee koko pelin tilaa yksinoikeudella: kaikki tieto lähetetään sen käsiteltäväksi, se suorittaa pelilogiikan ja lähettää jatkuvasti pelaajille pelin tämänhetkisen tilan. Tämä tekee huijaamisesta vaikeampaa, sillä asiakkaan ohjelma ei voi suorittaa itsenäisesti toimintoja, esimerkiksi tappaa vihollisen tai muuttaa pelistä saatuja pisteitä. (Unity Technologies 2014b.) Autoritaarisuus tarkoittaa kuitenkin sitä, että palvelimen vastuulla on paljon suoritettavia käskyjä. Myös tietoa tarvitsee lähettää enemmän palvelimen ja asiakkaiden välillä. Ei-autoritaarisessa palvelimessa tilanne on päinvastainen: itse pelilogiikka suoritetaan jokaisen asiakkaan pelissä, palvelin pelkästään synkronoi jokaisen pelin tilan samaksi ja asettaa käskyt suoritettaviksi. (Unity Technologies 2014b.)

Ei-autoritaarinen palvelin on myös kehittäjän näkökulmasta helpompi toteuttaa, koska silloin ei tarvitse välttämättä tehdä pelin tilaa ennakoivia menetelmiä. Palvelimen ja asiakkaan välillä voi olla viivettä, jolloin pakettien[1] vastaanottamisessa ja käsittelyssä kestää aikaa. Tätä paketin lähettämisen ja vastaanottamisen kestävää aikaa kutsutaan latenssiksi. (Rouse & Blair 2014.) Yhteyden jumiutuessa se voi olla hyvinkin korkea. Lisäksi palvelimen ja asiakkaan välinen fyysinen sijainti vaikuttaa siihen. Latenssi täytyy ottaa autoritaarisessa palvelimessa paremmin huomioon, joskin lähiverkossa latenssi on olematon.

1 Paketit sisältävät verkon välityksellä lähetettävän tiedon.

6.2 Laitekohtaiset peliversiot

Koska PC vaatii erilaisia ominaisuuksia kuin mobiililaitteet, täytyy pelistä tehdä kaksi eri versiota. Näiden eri versioiden välinen laajuus riippuu luonnollisesti pelinkehittäjän valitsemista muista menetelmistä. Vastuuta pelin toiminnallisuudesta voidaan siis jakaa PC:n ja mobiililaitteiden välillä. Joka tapauksessa vain PC-versio sisältää palvelimen toiminnallisuuden.

Unity-pelimoottori mahdollistaa ohjelmakoodin kääntämisen eri laitteille. On myös mahdollista määritellä, mitä lähdekoodia eri laitteille käännettävissä versioissa käytetään. (Unity Technologies 2014c.) Mobiiliversiosta voitaisiin siis jättää pois PC:n pelinäkymä ja kommunikaatioon liittyviä komponentteja. PC-versiosta voidaan sulkea pois mobiililaitteelle suunnatut ominaisuudet. Tällä tapaa pystytään kehittämään kahta tai useampaa versiota pelistä yhtäaikaisesti, jolloin projektin hallinta helpottuu. PC- ja mobiiliversioita ei siis tarvitse kehittää kokonaan erikseen, vaan se onnistuu rinnakkain.

Eri peliversioiden etu on se, että eri laitteille voidaan valita juuri ne ominaisuudet ja toiminnallisuudet, jotka on tarkoitettu. Vähimmäisvaatimus mobiilipuolen peliohjelmalle on ottaa yhteys palvelimeen, näyttää kuva näytöllä ja lähettää viestejä palvelimelle. Myös itse peli on mahdollista suorittaa mobiililaitteessa, jolloin se käyttää omaa laitteistoaan pelin laskutoimituksille.

6.3 Verkko-ominaisuuksien hyödyntäminen

TCP/IP on protokolla eli joukko sääntöjä, jotka määrittelevät, kuinka kommunikaatio tietokoneiden välillä tapahtuu. TCP eli Transmission Control Protocol vastaa nimensä mukaisesti tiedon lähettämisestä, tarkalleen ottaen viestin kokoamisen paketeiksi. Nämä paketit sisältävät tiedot, jotka halutaan lähettää tietokoneesta toiseen. TCP ottaa myös paketin vastaan. IP eli Internet Protocol taas vastaa paketin ohjaamisesta oikeaan osoitteeseen. (Rouse 2008.) Protokollan mukaisesti käytetään IP-osoitteita, jotka ovat 32- (IPv4) tai 128-bittisiä (IPv6) binäärilukuja. IPv4-osoitteet ovat muotoa 198.60.49.28, kun taas Ipv6-osoitteet ovat muotoa 412a:74e4:552c:1d9f:7521:35c1:325b:9a52. (ARIN 2014.) UDP eli

User Datagram Protocol on toinen protokolla tiedon lähettämiseksi. Se on nopeampi vaihtoehto TCP:lle, joskin se ei ole tarkka pakettien saapumisjärjestyksestä. (Rouse 2005.)

IP-osoitteen lisäksi kommunikaatiossa käytetään portteja, jotta yhteyttä voidaan käyttää yhtä aikaa moneen eri tarkoitukseen. Portti on yksinkertaisesti numero, jonka arvo on välillä 0–65536. Esimerkiksi ohjelmat tarvitsevat oman vapaan portin, jotta ne toimivat Internetissä. Portit varmistavat siis, että tieto ei mene sekaisin eri sovellusten välillä. (Abrams 2004.) Portit 0–1023 ovat järjestelmäprosessien käytössä (IANA 2014). Numerosarjassa 251.60.49.28:5313 ensimmäinen osa on IP-osoite, toinen osa kuvaa porttia.

Tietoa on mahdollista lähettää laitteiden välillä eri protokollia käyttäen. Esimerkiksi Wii U -konsoli käyttää GamePadissaan omaa protokollaa. WLAN on kotona toimiva langaton lähiverkko, jonka kautta verkko-ominaisuuksia voidaan hyödyntää helposti Wi-Fi-laitteiden avulla. Joka tapauksessa TCP/IP:hen tai UDP/IP:hen perustuva kommunikaatiomalli on varsin helppo toteuttaa, sillä se ei vaadi tueksi erillisen laitteiston tai ohjelmiston suunnittelemista kommunikaatiota varten (olettaen, että käytettävissä on vähintään lähiverkko).

6.3.1 Peliin yhdistäminen

Jotta pelaaja voi liittyä mukaan peliin, tarvitaan palvelimen IP-osoite ja portti (Unity Technologies 2014b). Master Server on Unityn oma järjestelmä, eräänlainen ”aula” peleille, jonka kautta voi löytää palvelimen IP-osoitteen pelille (Unity Technologies 2014d). Tämä ei ole välttämätön peliin liittymisen kannalta, sillä IP-osoite voidaan syöttää manuaalisesti yhteyttä muodostaessa. Se on helpoiten toteutettava tapa yhdistää asiakas palvelimeen. Käytettävyyden kannalta automaattinen yhteyden muodostaminen on kuitenkin huomattavasti parempi vaihtoehto. Silloin voidaan käyttää edellä mainittua Master Serveriä tai UDP-lähetystä.

Unity Master Serverin kohdalla täytyy huomioida, että palvelin täytyy pystyttää itse. Pelit käyttävät tätä palvelinta yhteyden muodostamisessa. (Unity Technologies 2014e.) Unityllä on testikäyttöä varten Master Server -palvelimia, mutta niitä ei ole tarkoitettu tuotantokäyttöön. Koska Master Serverin ylläpidossa peliä varten on oma vaivansa, sitä ei tässä prototyypissä tulla käyttämään.

UDP-lähetyksellä on mahdollista löytää IP-osoite ja yhdistää laite verkkoon. Tämä tapahtuu C#-kielen verkko-ohjelmointikirjastoja hyödyntäen, sillä Unityllä itsessään ei ole tukea UDP-lähetykseen. Tarkalleen ottaen laite voi lähettää tiedon verkkoon IP-osoitteestaan, jonka kuuntelija voi ottaa talteen. (Unity Answers 2013.)

6.3.2 Unityn verkkokirjastot

Unity käyttää kahta tapaa kommunikoida asiakkaan ja palvelimen välillä. Etäproseduurikutsut (RPC eli Remote Procedure Call) suorittavat jonkin tietyn funktion eli toiminnon ohjelmasta verkon välityksellä. Niitä käytetään muun muassa tapahtumia varten. (Unity Technologies 2014b.) Esimerkiksi palvelin voi lähettää RPC:n, jossa tuhotaan kaikkien asiakkaiden pelistä Pelaaja 1:n hahmo. Pelikoodin suoritus tapahtuu paikallisesti asiakkaan ohjelmassa (Unity Technologies 2014b).

Tilasynkronisaatio (State Synchronization) on nimensä mukaisesti tapa synkronoida jokin osa pelitilasta asiakkaiden välillä. Tiedot lähetetään jatkuvasti verkon välityksellä kaikille asiakkaille – esimerkiksi jokaisen pelaajan sijainti pelimaailmassa täytyy olla kaikkien tiedossa. Koska tiedon lähettäminen on reaaliaikaista, se vie jatkuvasti verkon kaistaa. Reliable Delta Compressed on yksi tapa synkronoida pelin tila Unityssä. Se pitää tietoja jatkuvasti silmällä ja lähettää paketteja pelkästään tietojen muuttuessa. Unreliable on toinen tapa synkronointiin. Silloin ei tarkisteta, onko paketti tullut perille. Tämän takia kaikki tiedot lähetetään kerralla pelkästään muuttuneiden tietojen sijaan. (Unity Technologies 2014f.)

Unity 4 käyttää RakNet-kirjastoa verkko-ominaisuuksien toteuttamiseen. Tästä seuraa, että pakettien lähetys tapahtuu UDP/IP:llä (RakNet 2014b). Siitä huolimatta Unity voi varmistaa, että jokainen paketti tulee perille. Paketti lähetetään uudelleen, kunnes se vastaanotetaan. Muut paketit odottavat sillä välin jonossa. (Unity Technologies 2014f.)

Sekä etäproseduurikutsut että tilasynkronointi tapahtuu Unity-pelimoottorissa NetworkView-komponenttien[1] avulla. Ne voivat tarkastella peliobjektin[2] tilan muutoksia, joiden tieto voidaan lähettää asiakkaille. Jokaisella NetworkView-komponentilla on tunnus (ID). Tämän avulla jokainen paketti voidaan ohjata tiettyyn komponenttiin. (Unity Technologies 2014g.)

Verkko-ohjelmointi perustuu soketteihin. Ne ovat tapa kommunikoida asiakkaan ja palvelimen välillä, eräänlaisia yhteyden päätepisteitä. (Rouse & Harschutz 2005.) Korkeamman tason verkkokirjastoja käyttäessä (esimerkiksi RakNet) ohjelmoija ei itse tarvitse soketteja, sillä kirjasto käyttää niitä pelkästään sisäisesti.

6.4 Paikallisesti suoritettujen pelien synkronointi

Tilasynkronoinnin ja etäproseduurikutsujen avulla asymmetrinen pelikokemus voidaan toteuttaa siten, että palvelin lähettää mobiililaitteisiin tarvittavat tiedot, esimerkiksi pelaajien ja vihollisten sijainnit (kuva 18). Pelinkehittäjän näkökulmasta voidaan valita, kuinka paljon palvelin käsittelee tietoa eli autoritaarisuuden määrää. Tässä tapauksessa pääasiassa ei-autoritaarinen palvelin toimii hyvin ja on helpompi toteuttaa. Mobiililaitteilla on omaa laskentatehoa, jota voidaan käyttää pelin 3D-grafiikan ja pelilogiikan suorittamiseen. Näiden laitteiden suorituskyky vaihtelee kuitenkin suuresti, joten siihen ei voida luottaa täysin. Mobiililaitteiden laskentakyky on huomattavasti rajatumpi kuin PC:n.

1 Komponentit ovat ohjelmoinnissa itsenäisiä kokonaisuuksia, eräänlaisia ”palikoita”, joilla on omistussuhde johonkin toiseen rakenteeseen. Unity hyödyntää toteutuksessaan peliobjekti-komponentti-mallia, jolloin peliobjektit sisältävät komponentteja.

2 Peliobjekti on pelinkehityksessä käytetty ”perusyksikkö”. Jos jotakin halutaan asettaa peliin, se on silloin peliobjekti. Esimerkiksi pelaaja ja viholliset ovat peliobjekteja.

Kuva 18. Pelitilan synkronointi PC:n ja mobiililaitteen välillä.

Tämän toteutustavan hyvä puoli on kuitenkin se, että Unity tukee verkko-ohjelmointia suoraan kirjastoissaan. Toisaalta taas voidaan tarvita suunnittelua, että palvelimen ja asiakkaan välinen kuormitus on juuri sopiva. Pelinäkymät voivat olla laitteissa erinäköiset PC:llä ja mobiililaitteella, joten tieto ei välttämättä sovi yhteen niiden välillä. Peli voidaan tietenkin suunnitella niin, että se on melko identtinen kummallakin laitteella. Esimerkiksi PC:llä kameran kuvakulma on ensimmäisen persoonan perspektiivissä ja peligrafiikka yksityiskohtaisempaa, mobiililaitteella ylhäältä kuvattu ja typistetty näkymä vie vähemmän laskentatehoa. Se asettaa kuitenkin rajoituksia pelin suunnittelulle – täysin erilaisia tilanteita pelistä voi olla vaikea toteuttaa.

6.5 Streaming

Streaming tarkoittaa tiedon lähettämistä reaaliaikaisena virtana. Esimerkiksi videota voidaan ”striimata” palvelimelta tietokoneelle. Se on yksi vaihtoehto asymmetrisyyden toteuttamiseksi. Palvelin hoitaa siis kaiken pelin laskennan ja lähettää pelkän kuvan mobiililaitteisiin (kuva 19). Kolmiulotteisissa peleissä kamera määrittelee, mikä alue pelitilanteesta näytetään ruudulla. Periaate on siis samantapainen kuin millä tahansa muullakin kameralla. Kameran sisältö on mahdollista piirtää ruudulle tai kuvaan (pelinkehityksessä puhutaan tekstuureista) Render Texture -ominaisuudella (Unity Technologies 2014h).

Kuva 19. Tekstuurin lähettäminen palvelimelta mobiililaitteeseen.

Streaming tapahtuisi pelissä asettamalla palvelimen toisen kameran näkymä tekstuuriin, joka lähetetään verkon kautta eteenpäin mobiililaitteeseen. Tässä tapauksessa joudutaan myös hyödyntämään verkko-ominaisuuksia, mutta palvelimen täytyy lähettää pelistä pelkkä kuva asiakkaalle. Ennen kuvan lähettämistä täytyy huomioida mobiililaitteen resoluutio eli kuinka monta pikseliä[1] ruudulla on. Erityisen tärkeä on kuvasuhde (eli kuinka leveä kuva on korkeuteen nähden), sillä lähetettävän kuvan tulisi täyttää koko mobiililaitteen näyttö. Täyttö voidaan toteuttaa asettamalla kameran koko mobiililaitteelle sopivaksi tai skaalaamalla kuva oikean kokoiseksi. Jälkimmäinen tapa aiheuttaa kuitenkin ongelmia kuvasuhteen ollessa yhteensopimaton mobiililaitteen kanssa.

Tekstuurista voidaan hakea tietokoneen näytöllä olevien pikselien sisältämä väritieto (kuva 20). Tämä on siis kaikki se tieto, joka täytyy tässä tapauksessa lähettää palvelimelta asiakkaalle. Kun paketti saapuu perille verkon välityksellä, pikselien väritiedoista voidaan asiakkaan näytölle luoda tekstuuri. Jatkuvasti lähetettynä tämä luo illuusion siitä, että asiakas olisi pelissä mukana.

Asiakkaan toinen rooli on lähettää kaikki mobiililaitteen syötteet palvelimelle, jossa ne käsitellään. Mobiilipelaaja voi esimerkiksi koskettaa jotakin kohtaa näytöstä, jolloin pelin hahmo reagoi liikkeeseen.

1 Pikselit ovat näytöllä olevia värillisiä pisteitä, joista kuvat muodostuvat.

Kuva 20. Pikselien väritieto ruudulla ja taulukossa tietokoneen muistissa.

Miten streaming olisi mahdollista toteuttaa ohjelmallisesti? Tiedon lähettämisessä voidaan käyttää joko etäproseduurikutsua tai omaa lähetystapaa soketeilla TCP:tä tai UDP:tä käyttäen. Ongelmaksi tässä voi osoittautua tiedon muoto ja määrä. Protokollilla on omat rajoituksensa paketin yksikön koon suhteen: tätä kutsutaan nimellä MTU eli Maximum Transmission Unit (Webopedia 2014). Olisi tärkeää huomioida, että tieto pakataan tiiviiseen muotoon, jotta se voidaan lähettää jatkuvana virtana palvelimelta asiakkaalle.

Tiedon muoto vaikuttaa lähettämistapaan. Väritiedot voidaan hakea tekstuurista GetPixels-funktion avulla. Toisaalta tekstuuri voidaan muuttaa binäärimuotoon[1] EncodeToPNG- tai EncodeToJPG-funktiolla. ULIB on Unityn lisäosa, joka tarjoaa mahdollisuuden muuttaa tekstuurin binääritaulukoksi. (Unity Answers 2011.) ULIBin avulla on itse asiassa mahdollista muuttaa mikä tahansa tieto binääritaulukoksi tai tekstimuotoiseksi merkkijonoksi. Näistä jälkimmäinen muoto on mahdollisesti helpompi käsitellä ohjelmallisesti. Tieto voidaan lähettää binääritaulukossa tai merkkijonona etäproseduurikutsun avulla. OnSerializeNetworkView-funktio ei hyväksy näitä muotoja. Sokettien käyttäminen vaatii ylimääräistä ohjelmoimista, joten jos on mahdollista lähettää väritiedot Unityn verkkokirjastojen toiminnallisuuden kautta, toteuttaminen olisi helpompaa.

1 Binäärimuodossa tieto koostuu tavuista, jotka ovat 8-numeroisia binäärilukuja. Tarkoitan pääasiassa binääritaulukkoa, joka voi sisältää useita tavuja.

Tekstuurin rakentaminen uudelleen tapahtuu asiakkaan pelissä vastaanotetun tiedon perusteella. Väritiedoista voidaan Unityssä muodostaa uusi tekstuuri SetPixels- tai LoadImage-funktiolla. Ensimmäinen näistä käyttää väritietoja, toinen binääritaulukkoa. Rakennettu tekstuuri asetetaan lopuksi mobiilipelaajan näkymään.

6.6 Kriteerit asymmetrisyyden toteuttamiselle

Asymmetrisyyden toteuttaminen lähettämällä kuva olisi helppoa, koska itse pelitietoja ei tarvitse synkronoida laitteiden välillä. Palvelin toimisi siis autoritaarisesti. Mobiililaite tarvitsee periaatteessa vakiomäärän laskentatehoa, jotta se voi vastaanottaa ja näyttää kuvan ruudulla. Syötteiden lähettäminen ei todennäköisesti kuormita asiakasta paljon. Mitä enemmän pelaajia on, sitä enemmän kuormaa toisaalta asetetaan PC-pelaajan tietokoneelle.

Kuvan lähettäminen voi kuitenkin aiheuttaa suuria ongelmia. Jos tavoitteena on lähettää 60 kuvaa sekunnissa mobiililaitteeseen, tiedon lähettämisen täytyy tapahtua varsin nopeasti. 30 ruutua sekunnissa olisi kuitenkin vielä hyväksyttävä (tosin ei toivottava) nopeus ruutupäivityksille. Pikselitiedot veisivät sellaisenaan paljon kaistaa, esimerkiksi 1440 x 900 kokoisessa tekstuurissa on 1 296 000 pikseliä. Tämä pikselimäärän käsittely voi viedä myös prosessorilta huomattavasti aikaa. Lähiverkossa lähetysnopeus on suuri, mutta edellä mainitut pakettien tietomäärän rajoitukset voivat hidastaa ja vaikeuttaa tätä prosessia huomattavasti. Jos kuvan lähettäminen osoittautuu epäluotettavaksi, asymmetrisyys kannattaa toteuttaa toisella tavalla.

Hyvässä asymmetrisessä pelikokemuksessa pelaaja pystyy olemaan mukana pelissä reaaliaikaisesti. Viivettä pelitilojen välillä ei siis saa olla paljon. PC:n ja mobiililaitteen tulisi pystyä kommunikoimaan keskenään jatkuvasti. Jos esimerkiksi mobiililaitteen pelaaja painaa kosketusnäyttöä, tulisi pelaajahahmon liikkeen näkyä PC:llä ja mobiililaitteella. Pelitilojen välille ei saa siis tulla liian suurta eroa (näytöllä oleva kuva ei vastaa pelin oikeaa tilaa tarpeeksi tarkasti).

Tavanomaisessa verkko-ohjelmoinnissa sulava yhteys on mahdollista toteuttaa, kun huomioidaan lähetettävän tiedon määrä. Vaikka pakettien lähetysnopeus ei olisi korkea, on mahdollista ennakoida esimerkiksi pelaajan liikkeitä. Tämä ei onnistu silloin, kun tietokoneelta lähetetään mobiililaitteeseen pelkkä kuva. Se vaatii jatkuvasti säännöllisen pakettien lähetysnopeuden, jotta pelitilanteesta muodostuva kuva on riittävä asymmetrisen kokemuksen tuottamiseksi. Esimerkiksi 1 kuva sekunnissa muistuttaisi enemmän hitaasti päivittyvää verkkokameraa. Toisaalta tilasynkronoinnin puolesta puhuu se, että pakettien lähetysnopeus voi olla korkea lähiverkossa. Pelikokemuksen täytyy ehdottomasti olla koherentti verkko- ja asymmetrisissä peleissä, koska muuten pelin mielekkyys kärsii liikaa ja muuttuu hajanaiseksi. Lisäksi tuki jopa neljälle yhtäaikaiselle pelaajalle olisi tarpeellinen, jotta peliä voisi pelata useampi kuin vain kaksi pelaajaa kerrallaan.

7 Prototyypin toteutus

Peliprototyyppi on alkukantainen toteutus pelistä, jota ei ole tarkoitus julkaista. Niitä hyödynnetään usein peli-ideoiden arvioinnissa. Tämän prototyypin kehitys tapahtui mobiilipuolella pääasiassa low-end-laitteilla. Puhelimena käytössä oli Sony Xperia Tipo (2012, 320x480, Android 4.0.4) sekä tablet-tietokoneena Lenovo IdeaTab A1000L-F (2013, 1024x600, Android 4.1.2). Ne edustavat tällä hetkellä todennäköisesti keskiarvoa alhaisempaa laskentatehoa. Laskennallisen tehon vähäinen määrä ei ole mobiililaitteella prototyypin toteutukselle suuri haitta tai este. Tässä prototyypissä ei keskitytty laitteiden välisiin eroihin tai pelin suorituskykyyn, sillä se on tärkeämpää oikeassa tuotannossa (lisää syitä tälle on tarkemmin seuraavassa kappaleessa; niiden vuoksi laitteet ovat myös sopivat työn toteutusta varten). PC on pelikäyttöön tarkoitettu pöytätietokone (AMD Phenom II X4 B55 (2010), AMD Radeon HD 7850 (2012)). Se edustaa tehokasta PC:tä, jonka laskentateho on korkeampi kuin esimerkiksi tyypillisten kannettavien tietokoneiden. Prototyyppiä on myös testattu toimeksiantajan toimesta muun muassa Applen laitteilla (iPad/Mac). Lähiverkko on muodostettu NetGear N300 WiFi-adapterilla, jonka teoreettinen nopeus on jopa 300 megabittiä sekunnissa.

Prototyypin pääasiallinen tarkoitus on saada aikaan asymmetrinen pelikokemus PC:n ja mobiililaitteiden välille. Pelattavan prototyypin avulla on mahdollista saada empiiristä tietoa asymmetristen pelien toiminnasta näiden laitteiden välillä: se voidaan nähdä ja kokea. Prototyyppi pyrkii vähimmäisyydessään sisältämään yhteyden PC:n ja usean mobiililaitteen välillä pelitilojen synkronointia hyödyntäen sekä laitteiden perusteella määritellyt roolit. Toisin sanoen pelaajien täytyy pystyä liikkumaan saman maailman sisällä ja nähdä se hieman eri näkökulmasta. Pelaajien roolitusta käsitellään tarkemmin myöhemmissä luvuissa. Lisäksi toteutettiin yksinkertainen streaming-menetelmä, jolla on mahdollista lähettää pelistä kuva verkon välityksellä PC:ltä mobiililaitteeseen. Se on eräänlainen koe, jolla on mahdollista nähdä onko menetelmä missään muodossa mahdollinen Unity-pelimoottorissa.

Tämän asymmetrisen peliprototyypin kehityksellä ei pyritty optimaalisiin ratkaisuihin. Tärkeämpää on nähdä, miten asymmetrisyys toimii näillä laitteilla. Siispä tässä työssä ei keskitytä myöskään eri laitteille optimointiin tai laitteiston asettamiin rajoitteisiin. Pelattavaa ei varsinaisesti tarvita paljon. Roolien ja asymmetrisyyden esittelyn kannalta on kuitenkin merkittävää, että peli-idea on täytetty. Itse peliä käsitellään tarkemmin tämän luvun aikana.

7.1 Pelaajien näkymät

Koska pelin ideana on toteuttaa kaksi erilaista näkymää, ne täytyy määritellä hieman tarkemmin. PC-pelaajan on siis tarkoitus pelata ensimmäisen persoonan näkökulmasta perinteisen ammuskelupelin tyyliin. Pelityyppiä kutsutaan yleisesti lyhenteellä FPS eli First Person Shooter. Suurin tunnusmerkki on pelaajan ase, joka on pelin keskipisteessä (kuva 21). Itse pelihahmosta näkyy vain kädet. Pelaaja voi katsella ympärilleen liikuttamalla hiirtä ja ampua hiiren painikkeilla. Liikkuminen tapahtuu W, A, S ja D -näppäimillä. Kentällä on vihollisia, jotka taistelevat pelaajaa vastaan. Aseen tähtäin on tyypillisesti pysyvästi ruudun keskellä. Käyttöliittymä on varsin yksinkertainen – siinä voi olla tietoja esimerkiksi pelaajan elinvoimasta, aseesta sekä ammusten määrästä.

Kuva 21. Tyypillinen FPS-näkymä, kuvankaappaus Command & Conquer Renegade -pelistä (2002).

Ylhäältä päin kuvatuissa ammuskelupeleissä (top-down shooter) pelinäkymä on laajempi (kuva 22). Kamera seuraa pelaajan liikkeitä automaattisesti ylhäältä käsin. Liikkuminen voidaan toteuttaa kolmannessa persoonassa joko kosketuspohjaisesti tai näppäinpohjaisesti. Pelaaja liikkuu siis joko näppäinten osoittamaan suuntaan tai osoittimella painaistuun kohtaan. Hahmon täytyy jälkimmäisessä tapauksessa osata automaattisesti löytää oikea reitti. Toimenpidettä kutsutaan reitinetsinnäksi. Hahmon täytyy myös kääntyä suuntaan, johon liikutaan tai ammutaan.

Tähtääminen voi tapahtua esimerkiksi hiiren osoitinta seuraamalla. Kehitettävässä prototyypissä täytyy huomioida kosketusnäytön käyttäminen – hiirellä toteutettava liikkuminen poikkeaa siitä huomattavasti. Mobiililaitteella liikkuminen voi tapahtua kosketuksen tai virtuaalisen ohjaustatin avulla. Merkittävä vaatimus mobiilipelaajan kannalta onkin erilainen näkymä, jonka mahdollistaa pelaajaa yläpuolelta seuraava kamera. Näkymä voi olla koko ajan tietyn kokoinen tai pelaajalle voi antaa mahdollisuuden zoomata tai kääntää sitä. Lisäksi erittäin tär-

keä on hahmon hallinnan toteuttaminen kosketusnäytön avulla.

Kuva 22. Kuvankaappaus yhteistoiminnallisesta Alien Swarm -pelistä (2010).

Toisin sanoen mobiilipelaajan täytyy pystyä navigoimaan pelissä kosketusnäytön avulla. Kameran toiminnan tulisi olla automaattista – tosin sitäkin voi olla mahdollista hallita. Mobiilipelaajan tulisi pystyä ampumaan aseilla samalla tapaa kuin PC-pelaajan. Prototyypissä olisi tarkoitus, että mobiilipelaajat ovat avustajia ja tekevät vähemmän vahinkoa.

7.2 Character controller -ratkaisut

Pelinkehityksessä hahmoa hallitsevaa komponenttia kutsutaan nimellä character controller. Sen päämäärä on liikuttaa hahmoa paikasta toiseen sekä hallita mahdolliset törmäykset. Hahmon ei tulisi myöskään päästä esimerkiksi seinien läpi. Character controllerilla voi olla monia ominaisuuksia, esimerkiksi hyppääminen, juokseminen, mäkien nouseminen ja laskeminen. Unity tarjoaa muutamia mahdollisuuksia character controllerin toteuttamiseksi. Unityn valmis character controller -komponentti sisältää toiminnot hahmon liikuttamiseksi sekä

törmäysten hallinnan. Työssä käytetään tätä valmista komponenttia, koska silloin voidaan keskittyä tavallisen liikkumisen kehittämisen sijaan hahmon toimintoihin. Erityisesti prototyyppiä tehdessä on hyvä pyrkiä säästämään aikaa, sillä kehitysaika on lyhyt.

Kolmannen persoonan character controller voidaan toteuttaa Unityn sisäänrakennetun navigaatiojärjestelmän avulla (NavMesh). Liikkuakseen tiettyyn pisteeseen peliruudulla tarvitaan jonkinlaista reitinetsintää – tähän NavMesh on toimiva ratkaisu. Se luo etukäteen pelikentän perusteella alueet, joilla peliobjektit voivat liikkua. Tätä järjestelmää voivat käyttää niin pelaajat kuin tekoälyn ohjaamat vihollisetkin: se tukee myös dynaamisia eli liikkuvia objekteja.

7.3 Pelaajien toteutus

Kuten edellisestä kappaleesta kävi ilmi, pelaajien liikkuminen täytyy eri laitteita hyödyntävässä asymmetrisessä pelissä toteuttaa alustasta riippuen. Toteuttamani toiminnallisuuden mukaan PC-pelaajan hahmoa liikutetaan näppäimistä saadun tiedon perusteella. Kameran sijainti taas asetetaan liikkumaan hahmon kohdalle. Kameraa käännetään hiiren liikkeen mukaan x- ja y-koordinaatistossa. Hiiren liikuttaminen oikealle pyörittää kameraa vaakatasossa hahmon ympäri, kun taas ylöspäin liikuttaminen kääntää kameraa taivasta kohti. Ongelmaksi ylös ja alas liikuttaessa osoittautui kameran pyörähtäminen ympäri. FPS-peleissä ei kuitenkaan pysty tähtäämään kuin korkeintaan ylös tai alas. Tämän takia kameran kulmalle täytyy asettaa rajat kokonaispyörähdyksen perusteella.

Kosketuksien hallintaan Unity tarjoaa melko yksinkertaisen toiminnon, jolla on mahdollista hakea näytöllä olevat kosketukset. Itse kosketuksien hallinta ja eleet täytyy toteuttaa silloin itse. Erityisen ongelmallista on kosketuksien testaaminen PC:llä, koska itse kosketusten haku toimii vain mobiililaitteilla. Asset Storessa on kuitenkin olemassa valmiita ratkaisuja kosketuksien ja syötteiden hallintaan. Jotta testaaminen onnistuu PC:llä suoraan, käytetään prototyypissä TouchScript-lisäosaa. Lisäksi se tekee eleiden hallitsemisen helpommaksi, sillä lisäosassa on valmiiksi määriteltyjä eleitä. Hahmon hallitsemiseen ei kuitenkaan

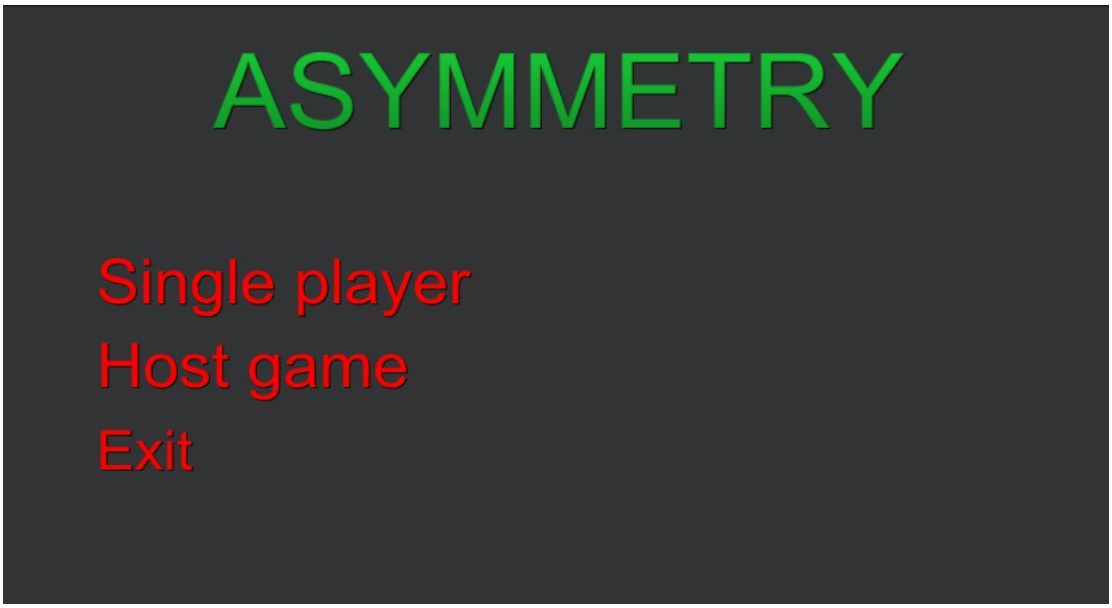
tarvita kuin yksinkertaista yhden näppäimen painallusta, jota myös Unity osaisi mallintaa hiiren avulla. Lisäosan erityinen etu on etenkin siinä, että se toimii varmasti kaikilla laitteilla, mikä pitäisi omassa järjestelmässä varmistaa erikseen.

Mobiilipelaajan hahmo liikkuu siis kosketuksen asettamaan paikkaan. Näyttöä voi painaa myös pohjassa, jolloin hahmo seuraa sormea jatkuvasti. Tämä tapahtuu siten, että kosketuksesta lähetetään maata kohti puolisuora, jonka leikkauskohta määrittelee liikkumiskoordinaatit. Tätä tekniikkaa kutsutaan nimellä raycasting. NavMesh-järjestelmä ottaa vastaan tämän koordinaatin, ja se osaa liikuttaa hahmon oikealle paikalleen esteistä huolimatta.

7.4 Käyttöliittymä

Päävalikko on olennainen osa peliä. Se on eräänlainen pelin aloituspiste. Tyypillisiä valikon toimintoja ovat yksinpeli, moninpeli, asetukset sekä pelistä poistuminen. Nämä kuuluvat asetuksia lukuun ottamatta myös toteutettavan pelin päävalikkoon (kuva 23). Prototyypin käyttöliittymät toteutettiin Unityn NGUI-lisäosalla, jonka avulla on mahdollista luoda käyttöliittymiä graafisesti. Työskentely on silloin intuitiivisempaa. Käyttöliittymien koko määrittyy halutessaan laitteen mukaan. NGUI vie vähän laitteistoresursseja, mikä on etu erityisesti mobiililaitteilla. Se on myös ominaisuuksiltaan huomattavasti monipuolisempi verrattuna Unity 4:n sisäänrakennettuun käyttöliittymäjärjestelmään. Sen avulla on esimerkiksi mahdollista luoda erilaisia efektejä käyttöliittymiin.

Käyttöliittymien välillä täytyy olla pieniä eroja PC- ja mobiililaitteiden välillä. Oikea käyttöliittymä aktivoidaan alustasta riippuen. Esimerkiksi PC-pelaajalla täytyy olla mahdollisuus aloittaa palvelinyhteys, kun taas mobiilipelaaja etsii palvelinta. Kun käyttäjät ovat yhdistäneet, vain PC-pelaaja voi aloittaa itse pelin.



Kuva 23. Peliprototyypin päävalikko. Punaiset tekstit ovat painikkeita.

7.5 Pelimekaniikan toteutus

Pelimekaniikka tarkoittaa kaikkea käytännössä tapahtuvaa pelin toiminnallisuutta. Tärkein osa pelimekaniikkaa prototyypissä on ampuminen. Vihollisia täytyy siis pystyä tuhoamaan ammuksilla. PC- ja mobiilipelaajalla tulisi kummallakin olla käytössä aseet. Ampuminen poikkeaa pelillisesti näkymissä huomattavasti toisistaan: Ensimmäisen persoonan ammuskelussa täytyy tähdätä hiirellä, jotta kohteisiin osuu. Tähtääminen voitaisiin myös toteuttaa ylhäältä kuvatuissa ammuskelupeleissä toisella tapaa, mutta pelin määrittelyssä ampuminen tapahtuu kohteita koskettamalla automaattisesti. Mobiilipelaajilla on kaksi huomattavaa etua PC-pelaajan nähden: suurempi näkymä sekä automaattinen ampuminen valittuihin kohteisiin. Tämä vaikuttaa myös mobiilipelaajan rooliin. Jotta vaikutus ei olisi ylivoimainen, mobiilipelaajat tekevät huomattavasti vähemmän vahinkoa kohteisiin kuin PC-pelaaja. Silloin avustavan pelaajan rooli säilyy, ja se korostaa pelin asymmetrisyyttä.

Ammuksen suunnan määrittely

PC-pelaajan ammuksen suunta määritellään näytön keskipisteen avulla, koska se on tähtäimen sijainti pelissä. Näytön keskipistettä kohti suunnataan puolisuora, jonka leikkauspiste toisen kohteen kanssa määrittelee suunnan sijainnin. Tämä suuntaa ammuksen tarkasti juuri siihen kohtaan, johon tähtäimen mukaan on tarkoitus osua. Aina kohdetta ei kuitenkaan ole tähtäimen kohdalla (esimerkiksi ilmaan ampuessa), jolloin suunta täytyy laskea erikseen tasoa vasten. Se on aina pelaajan osoittamassa suunnassa. Ammuksen suunta voitaisiin määritellä jokaisessa tapauksessa tasoa kohti leikkaavalla puolisuoralla, mutta silloin sen etäisyys pelaajasta vaikuttaa osumaan. Mobiilipelaajan ammuksen suunta taas määritellään paljon yksinkertaisemmin. Se tapahtuu laskemalla suuntavektori pelaajan ja kosketetun kohteen välillä sijaintikoordinaattien perusteella.

Ammuksien liike ja törmäykset

Peleissä on mahdollista määritellä, millaisella nopeudella ammukset kulkevat. Nopeus on todennäköisesti aika korkea, mutta ei välttämättä vastaa tosielämää. Liikkumisnopeus on kuitenkin merkittävä pelin kannalta, sillä törmäyksiä ei välttämättä havaita, jos ammus liikkuu yhden peliruudun aikana pitkän matkan. Unityn sisäänrakennettu järjestelmä ei havainnut kaikkia törmäyksiä. Koska ammukset ovat nopeita, törmäykset kohteiden kanssa on helppo määritellä puolisuoralla (raycasting). Prototyypissä toteutin tämän niin, että ensin tehdään lyhyen matkan testi. Jos kohde on lähellä, osuu ammus siihen nopeuden takia täydellä varmuudella, jolloin ylimääräisiä testejä ei tarvita. Jos kohde on kauempana, ammus tekee samaa testiä puolisuoralla yhä uudelleen ja uudelleen. Tämä ei ole ainoa tapa – puolisuoraa voi käyttää myös muulla tapaa. Halusin kuitenkin sellaisen toteutuksen, jossa ammusten lento kestää aikaa. Se ei tietenkään ole välttämätöntä. Ammuksissa on lisäksi Unityn TrailRenderer-komponentti, joka piirtää viivan ammuksen perään havainnollistaakseen ammuksen liikerataa pelaajalle.

Tuhottavat kohteet

Peleissä määritellään vahinko ja elinvoima numeroita käyttäen. Samaa tapaa hyödynnetään myös tässä prototyypissä. Jokaisella tuhottavalla kohteella on tietyn verran elinvoimaa (esimerkiksi 100). Ammukset vähentävät osuessaan tätä määrää. Kun elinvoiman määrä on 0, kohde tuhoutuu. Tekemäni CombatStatus-komponentti sisältää tiedot kohteen elinvoimasta sekä elossa olemisesta. Lisäksi jokaiselle kohteelle voidaan asettaa jokin toiminto, kun kohde ottaa vahinkoa tai kuolee. Jotta elinvoima olisi varmasti sama kaikilla laitteilla, palvelin toimii autoritaarisesti sen suhteen ja lähettää tiedon kaikille pelaajille.

Pelaajan syötteiden hallinta

Käsittelin luvuissa 7.1 ja 7.2 sitä, miten pelaajien liikkuminen tapahtuu. On kuitenkin hyvä mainita hieman tarkemmin, miten syötteiden hallinta toimii pelissä. Tätä varten suunnittelin peliin kolme erillistä komponenttia: FirstPersonController hallitsee nimensä mukaisesti PC-pelaajan hahmon syötteet ja toiminnot, TouchInputManager käsittelee TouchScript-lisäosan syötteet ja välittää ne ThirdPersonControllerille.

Ensimmäisen persoonan hallinta tapahtuu siis varsin itsenäisesti: jos painetaan hiiren painiketta, pelaaja ampuu. Mobiilipelaajien hahmot taas tarvitsevat tiedot kosketuksista, mikä tapahtuu erikseen TouchInputManager-komponentin kautta. Se käskee ThirdPersonControlleria esimerkiksi liikkumaan tiettyyn sijaintiin. Lisäksi se havaitsee myös kentällä olevat peliobjektit. Lisäksi First- ja ThirdPersonControllereissa on myös esimerkiksi viittaus[1] pelaajan käyttämään aseeseen, jolloin sen toimintoja voidaan kutsua.

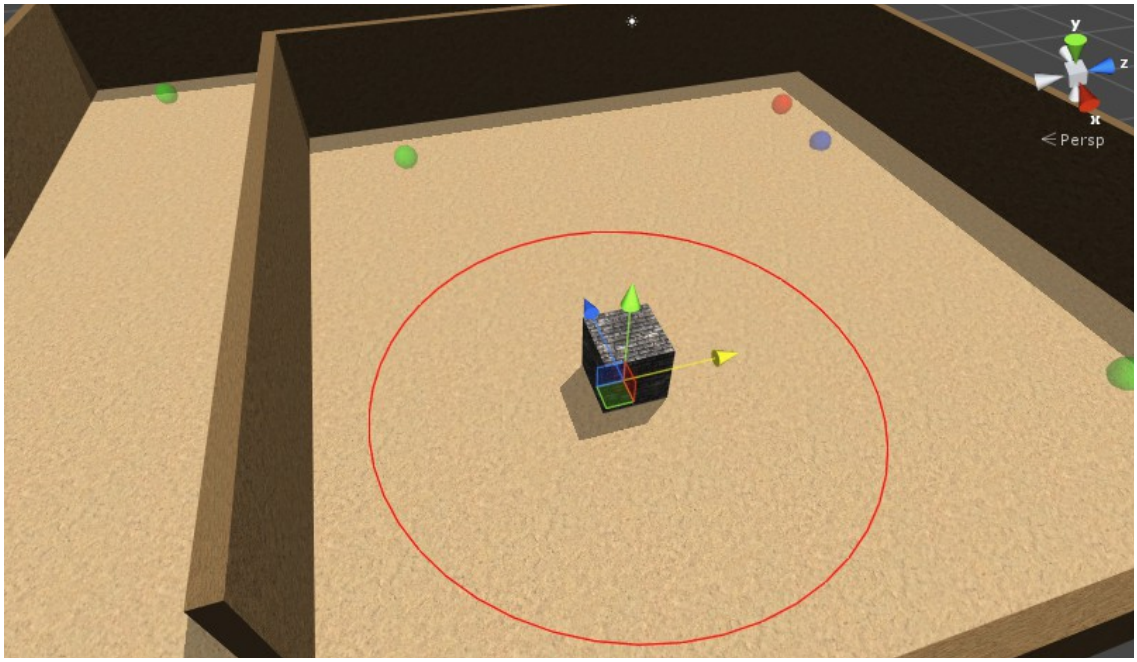
Mobiilipelaajan ampuminen eroaa myös PC-pelaajasta siinä, että sille on asetettu raja (etäisyys kohteesta). Jos pelaaja on kauempana, hahmo liikkuu rajalle asti. Jotta tämä oli mahdollista, kosketettu kohde täytyi tallentaa muistiin. TouchInputManager välittää kohteen ThirdPersonControllerille. Se pysyy kohteena niin kauan, kunnes se tuhotaan tai pelaaja liikkuu johonkin suuntaan.

1 Viittaus sisältää pelkästään muistiosoitteen. Sen avulla voidaan esimerkiksi käsitellä yhden peliobjektin muistia monesta eri paikasta kopioimatta sitä.

Viholliset

Kuten pelin määritelmään kuului, prototyypissä on ammuttavia vihollisia. Jotta pelikokemus olisi mielenkiintoisempi, näille vihollisille on toteutettu yksinkertainen tekoäly. Niiden toiminta kattaa seuraavat ominaisuudet: 1) Viholliset hyökkäävät lähimpänä olevan pelaajan kimppuun. Jotta vihollinen alkaa seurata häntä, sen täytyy olla tietyllä etäisyydellä. 2) Pelaajan täytyy olla näkyvissä, vihollinen ei voi nähdä pelaajaa seinän takaa. Tämän toteuttamiseksi hyödynnetään Unityn LineCast-toimintoa, joka tarkistaa onko kahden pisteen välillä peliobjekteja. 3) Lähietäisyydellä viholliset alkavat lyödä pelaajaa. Vihollisen täytyy olla kääntynyt pelaajaa kohti, että tämä on mahdollista. 4) Jos pelaajia ei ole lähietäisyydellä, viholliset liikkuvat luomisalueella. Pesän tuhoutuessa ne voivat liikkua vapaasti pelikentällä.

Vihollisten luomisalueet ovat tärkeä osa peliprototyyppiä. Sellainen on havainnollistettu kuvassa 24. Viholliset ilmestyvät pelikentälle tietyin väliajoin. Niihin on myös määritetty tietty maksimimäärä yhtäaikaisille vihollisille. Pelaajien tarkoitus on tuhota pesät, jotta vihollisia ei tule lisää. Ne ovat huomattavasti vihollisia kestävämpiä.

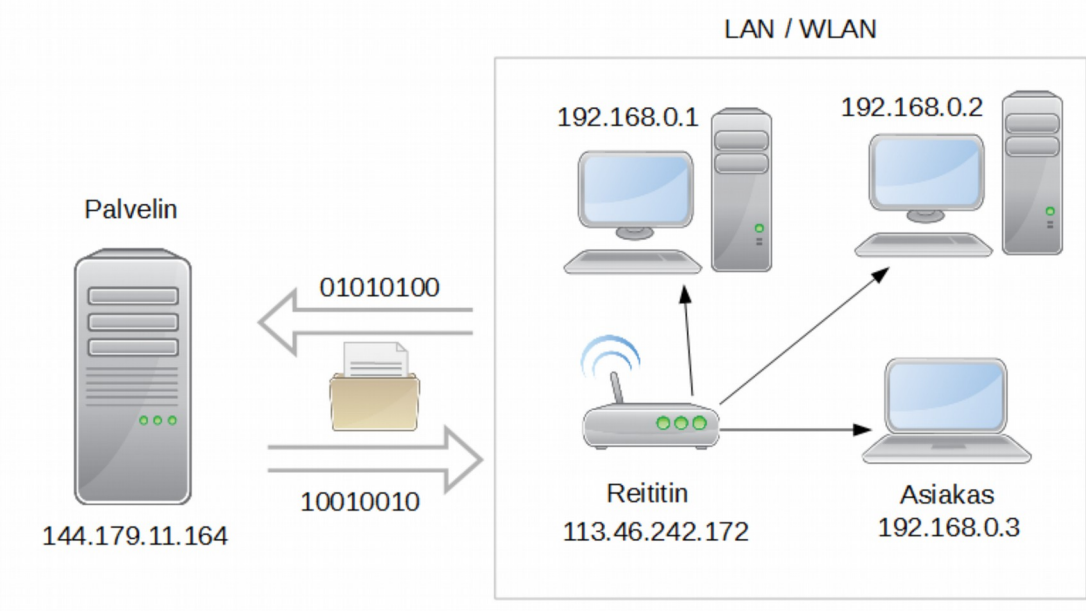


Kuva 24. Vihollisten luomisalue Unity-editorin näkymässä. Ympyrä merkitsee sen rajan.

7.6 Verkkoyhteyden luominen ja testaaminen

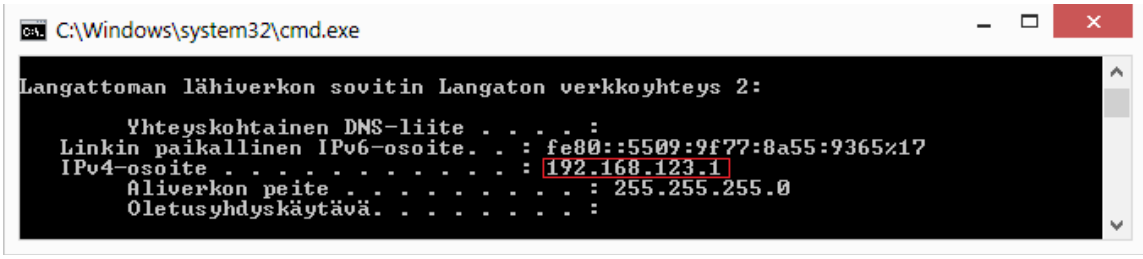
Tieto kulkee Internetissä kupari- tai valokuituja pitkin, jossakin tapauksessa jopa langattomasti radioaaltoja käyttäen. Viimeinen päätepiste ennen tietokonetta on reititin, joka on usein liitettynä kiinni tietokoneeseen verkkokaapelilla. (Titus 2009.) Reititin muodostaa lähiverkon (LAN eli local area network). Lähiverkkoon kiinnitetyt tietokoneet voivat kommunikoida keskenään ilman Internet-yhteyttä. (Rouse 2006.) Langattomassa lähiverkossa (WLAN) tietokoneet eivät tarvitse fyysisiä kaapeleita (Rouse 2010).

Reitittimellä on julkinen IP-osoite, jota kaikki samassa verkossa olevat tietokoneet voivat käyttää (kuva 25). Lähiverkossa laitteilla on yksityinen IP-osoite, johon ei voida suoraan ottaa yhteyttä ulkoapäin. (Unity Technologies 2014b.) Julkisen IP-osoitteen käyttö tapahtuu reitittimessä osoitteenmuunnoksen avulla (NAT eli Network Address Translation). Tämä lisää tietoturvaa ja vähentää tarvittavaa IP-osoitteiden määrää. NAT aiheuttaa kuitenkin ongelmia palvelimilla ja vertaisverkossa, sillä ulkopuoliset eivät silloin voi ottaa suoraa yhteyttä. On kuitenkin mahdollista ”aukaista” eli varata tietty portti tietylle laitteelle osoitteenmuunnoksesta huolimatta. (AfterDawn 2014.)



Kuva 25. Lähiverkon toiminta ja yhteys palvelimeen.

Ensimmäinen askel prototyypin toteuttamiseksi oli selvittää, miten verkkoyhteys asiakkaan ja palvelimen välille toteutetaan. Unityn verkkokirjasto on hyvin helppokäyttöinen, joten aiemman kokemuksen perusteelta yhteyden muodostaminen onnistui helposti. Network-luokasta löytyvät tähän tarvittavat toiminnallisuudet. Kuten mainitsin aiemmin, tarvitsin yhteyden muodostamista portin ja IP-osoitteen. Portiksi voi valita minkä tahansa vapaan numeron (oikealta väliltä). Muodostaakseni yhteyden samassa lähiverkossa oleville laitteille minun täytyi saada palvelimen yksityinen IP-osoite asiakkaita varten. Tämä onnistuu esimerkiksi Windowsin komentoriviltä komennolla ”ipconfig” (kuva 26).



Kuva 26. Yksityisen IP-osoiteen hakeminen ipconfig-komennolla.

Tosiasiassa tämä IP-osoite voi tietenkin vaihdella käytetystä laitteistosta riippuen. Sen takia mahdollisuus syöttää IP-osoite tai hakea se muulla tavalla on paljon parempi. Ideaalia olisi, että pelaaja voi automaattisesti hakea lähiverkon pelit, jolloin IP-osoitetta ei tarvitse syöttää erikseen.

Streamingiä varten ensimmäisessä pelinäkymässä (palvelin) on kaksi kameraa, toinen niistä on tarkoitettu Render Texture -ominaisuuden käyttöön. Asiakkaan pelinäkymässä on taas vain yksi pääkamera. Näkymään on määritelty myös suora taso, johon voidaan asettaa tekstuuri. Laitteiden välistä yhteyttä testasin asettamalla kumpaankin pelinäkymään kuution, joka liikkuu vasemmalta oikealle ja pyörii akselin ympäri. Lisäsin siihen NetworkView-komponentin, ja asetin Unityn tilasynkronaation käyttöön paikkatietoja varten. Liike oli hieman hidas asiakkaan pelissä, koska pakettien lähetysnopeuden vakioarvo on Unityssä 15. Arvon ollessa 60 liike oli sulavampaa. Lähiverkossa paketteja voidaankin lähettää paljon. Palvelimen toiminnallisuus on toteutettu NetworkHost-komponenttiin (listaus 1).

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

public class NetworkHost : MonoBehaviour
{
    // Here we store every player that connects and
    // the status of sending textures (true means occupied).
    // Note: only used for testing texture streaming.
    public Dictionary<int, bool> PlayerStatus { get; private set; }

    public static NetworkHost Instance;
    public bool IsRunning { get; private set; }


#if (UNITY_STANDALONE)

    public int listenPort = 1025;
    readonly int MAX_PLAYERS = 3;

    private IPSender _IPSender;
    private GameDataCommunication dataCommunication;

    void Start()
    {
        Instance = this;
        _IPSender = GetComponent<IPSender>();
        dataCommunication = GetComponent<GameDataCommunication>();
    }

    public void InitializeServer()
    {
        bool useNAT = !Network.HavePublicAddress();
        var error = Network.InitializeServer(MAX_PLAYERS, listenPort, useNAT);

        _IPSender.BroadcastIP();

        switch (error)
        {
            case NetworkConnectionError.NoError:
                break;
            default:
                Debug.Log("Error!");
                break;
        }
    }

    public void Disconnect()
    {
        Network.Disconnect();
        _IPSender.StopBroadcast();
    }

#region --- Server network events ---

    void OnServerInitialized()
    {
        IsRunning = true;
        Debug.Log("Server initialized!");
```

```
        PlayerStatus = new Dictionary<int, bool>();
    }

    void OnDisconnectedFromServer(NetworkDisconnection info)
    {
        IsRunning = false;
        Debug.Log("Server disconnected.");

        PlayerStatus = new Dictionary<int, bool>();
    }

    void OnPlayerConnected(NetworkPlayer player)
    {
        Debug.Log("Player connected!");

        PlayerStatus.Add(int.Parse(player.ToString()), false);
    }

    void OnPlayerDisconnected(NetworkPlayer player)
    {
        Debug.Log("Player disconnected!");
        Network.RemoveRPCs(player);
        Network.DestroyPlayerObjects(player);
        dataCommunication.RemovePlayer(player);

        PlayerStatus.Remove(int.Parse(player.ToString()));
    }

#endregion
#endif
}
```

Listaus 1. Palvelinyhteyden hallinta.

Jotta yhteyden muodostaminen olisi mahdollisimman helppoa, täytyy palvelimen IP-osoite lähettää lähiverkkoon. Kuvailin UDP-lähetyksen luvussa 6.2.1. Tätä varten oli saatavilla valmista lähdekoodia, jota käytin pohjana toiminnon toteuttamiselle. Ajatus oli seuraava: kun PC-pelaaja aloittaa palvelimen, se alkaa lähettää IP-osoitetta lähiverkkoon. Asiakas painaa omasta pelistään nappia, jolloin se alkaa etsiä IP-osoitetta. Jos asiakas löytää IP-osoitteen, mobiililaite yhdistetään palvelimeen. Pelaajan ei tarvitse siis huolehtia IP-osoitteen syöttämisestä. Lähetykselle ja vastaanottamiselle on kummallekin omat luokkansa[1], ohjelmakoodi on jaettu loogisesti kahteen osaan palvelimen ja asiakkaan välillä (listaus 2 ja listaus 3). Kohdat ”#if (Unity_STANDALONE)” ja ”#if ((UNITY_ANDROID || UNITY_IPHONE || UNITY_WP8))” merkitsevät alustakohtaisen lähdekoodin alkukohtaa.

1 Luokka toimii ohjelmoinnissa kuin pohjapiirustus. Sitä voidaan käyttää toistuvasti samanlaisten kokonaisuuksien luomiseen tai tiettyjen toiminnallisuuksien kutsumiseen. Se sisältää ohjelmakoodia. Myös komponentit ovat luokkia.

```
using UnityEngine;
using System.Collections;
using System.Net.Sockets;
using System.Net;
using System.Text;

public class IPSender : MonoBehaviour
{
#if (UNITY_STANDALONE)

    // Sender original source: http://answers.unity3d.com/questions/571052/net-
working-masterserver.html

    public int remotePort = 1026;

    UdpClient sender;

    public void BroadcastIP()
    {
        sender = new UdpClient(remotePort, AddressFamily.InterNetwork);
        IPEndPoint groupEP = new IPEndPoint(IPAddress.Broadcast, remotePort);
        sender.Connect(groupEP);

        InvokeRepeating("SendData", 0, 5f);
    }

    public void StopBroadcast()
    {
        CancelInvoke("SendData");
        if (sender != null) sender.Close();
        sender = null;
    }



    void SendData()
    {
        string customMessage = LocalIPAddress();

        if (customMessage != "" && sender != null)
        {
            sender.Send(Encoding.ASCII.GetBytes(customMessage),
            customMessage.Length);
        }

    }

    // Source: http://stackoverflow.com/questions/6803073/get-local-ip-address-c-
sharp
    public string LocalIPAddress()
    {
        IPHostEntry host;
        string localIP = "";
        host = Dns.GetHostEntry(Dns.GetHostName());
        foreach (IPAddress ip in host.AddressList)
        {
            if (ip.AddressFamily == AddressFamily.InterNetwork)
            {
                localIP = ip.ToString();
                break;
            }
        }
```

```
        return localIP;
    }

    void OnApplicationQuit()
    {
        StopBroadcast();
    }

#endif
}
```

Listaus 2. IP-osoitteen lähettäminen lähiverkkoon C#-sokettien avulla.

```
using UnityEngine;
using System.Collections;
using System.Net.Sockets;
using System.Net;
using System.Text;
using System;

public class IPReceiver : MonoBehaviour
{
 #if ((UNITY_ANDROID || UNITY_IPHONE || UNITY_WP8))

    // Receiver original source: http://answers.unity3d.com/questions/571052/net-
working-masterserver.html

    public int listenPort = 1026;

    NetworkConnection connection;

    UdpClient receiver;
    bool hasReceived = false;
    bool hasConnected = false;

    string IP = "";

    void Start()
    {
        connection = GetComponent<NetworkConnection>();
    }

    void Update()
    {
        if (hasReceived && !hasConnected)
        {
            string tmp = IP;
            IP = "Trying to connect...";
            connection.ConnectToServer(tmp);
            hasConnected = true;
            IP = "Connected to " + tmp;
        }
    }

    public void StartReceivingIP()
    {
        IP = "Searching for host...";
        try
        {
            if (receiver == null)
```

```
            {
                receiver = new UdpClient(listenPort);
                receiver.BeginReceive(new AsyncCallback(ReceiveData), null);
            }
        }
        catch (SocketException e)
        {
            Debug.Log(e.Message);
        }
    }

    private void ReceiveData(IAsyncResult result)
    {
        IPEndPoint receiveIPGroup = new IPEndPoint(IPAddress.Any, listenPort);
        byte[] received;
        if (receiver != null)
        {
            received = receiver.EndReceive(result, ref receiveIPGroup);
        }
        else
        {
            return;
        }

        string receivedString = Encoding.ASCII.GetString(received);

        if (!String.IsNullOrEmpty(receivedString))
        {
            IP = receivedString;
            hasReceived = true;

            receiver.Close();
            receiver = null;
        }
        else
        {
            receiver.BeginReceive(new AsyncCallback(ReceiveData), null);
        }
    }

    public void ClearIP()
    {
        IP = "";
        hasConnected = false;
        hasReceived = false;
    }

    void OnGUI()
    {
        GUIStyle style = new GUIStyle();
        style.fontSize = 16;
        style.normal.textColor = Color.black;
        GUI.Label(new Rect(10, Screen.height - 20, 100, 100), IP, style);
    }

    void OnApplicationQuit()
    {
        ClearIP();
        if (receiver != null) receiver.Close();
        receiver = null;
    }
```

```
#endif
}
```

Listaus 3. IP-osoitteen vastaanottaminen asiakkaan laitteessa.

7.7 Streamingin toteuttaminen

Lähdin toteuttamaan kuvan lähettämistä eli streamingiä saadakseni selville, onko se mahdollista Unityn ympäristössä. Lisäksi halusin tietää, miten hyvin se toimii. Prosessi ei kuitenkaan ollut aivan niin yksinkertainen kuin olin kuvitellut. Pikselitietojen hakeminen RenderTexture-tietotyypistä ei onnistunut suoraan. Pelkästään Texture2D-luokka tarjoaa mahdollisuuden tähän toimintoon. RenderTexturen muoto täytyi muuttaa kaksiulotteiseksi tekstuuriksi. Tämä voidaan toteuttaa ReadPixels-funktiolla, se kirjoittaa aktiivisen kameran tai RenderTexturen sisällön muistiin.

Parhaiten tekstuurin luominen onnistui luomalla RenderTexture ohjelmakoodissa ja asettamalla se aktiiviseksi kameraan. Seuraavaksi täytyi piirtää kameran sisältö tekstuuriin ReadPixels-funktiolla, kun itse pelin ruutu oli jo piirretty. Tälle Unity on määritellyt tapahtumafunktion nimeltä OnPostRender. Muistissa olevien pikseleiden arvot olivat muutoksen jälkeen oikeat.

Tämän jälkeen tekstuuri tulisi lähettää asiakkaalle. Miten sen voisi toteuttaa? Kun GetPixels-funktiota kutsutaan 60 kertaa sekunnissa, peli hidastuu todella huomattavasti. Tämä ei ole yllätys, koska pikseleitä on paljon. GetPixels32-funktio toimi kuitenkin nopeammin. Väritiedot piti muuttaa vielä binäärimuotoon tai merkkijonoksi, jotta ne voisi lähettää asiakkaalle. Kokeilin ULIB-kirjastoa tätä varten, mutta sekin hidasti peliä niin paljon, että se ei toiminut kunnolla. Suuren tietomäärän muuttaminen jatkuvasti toiseen muotoon vei liikaa prosessointitehoja.

Tekstuurin enkoodaaminen eli muuntaminen PNG-muotoon sai aikaan paljon paremman tuloksen. JPG-muoto oli kuitenkin vielä nopeampi, joten päätin käyttää lopuksi sitä. EncodeToJPG-funktion avulla sain muutettua tekstuurin binäärimuotoon, jolloin lähettäminen asiakkaalle on mahdollista etäproseduurikutsun

avulla. Se lataa tekstuurin asiakkaan muistiin ja asettaa sen aiemmin määriteltyyn tasoon pelinäkymässä.

Tulos ei kuitenkaan ollut täysin odotettu. Palvelin lähetti tietoa, mutta tekstuuri ei ilmestynyt asiakkaan ruudulle ollenkaan. Vasta tietojen lähetyksen loputtua kuva saattoi ilmestyä ruudulle. Oletin, että tietoa lähetetään käsiteltäväksi liian paljon kerralla. Siispä ratkaisin sen seuraavasti: palvelin lähettää tekstuurin vasta, kun asiakas ilmoittaa, että se on saanut tekstuurin ladattua ruudulle. Lähetys tapahtuu etäproseduurikutsulla. Tällä toimintamallilla kuva alkoi näkyä reaaliaikaisesti. Kuva ei ollut täysin sulava, mutta kohtalaisen toimiva kuitenkin.

Vaikuttaa siltä, että kuvien prosessointi vie huomattavasti laskentatehoa mobiililaitteella. LoadImage-funktio, joka ottaa vastaan ja lataa binääritiedot, on aika hidas. Sen nopeus riippuu luonnollisesti lähetetyn tekstuurin resoluutiosta sekä laitteen laskentatehosta. Esimerkiksi isonäyttöinen tablet-tietokone tarvitsee enemmän tehoa kuin pieninäyttöinen älypuhelin, koska tekstuuri on isompi. Silloin täytyy käsitellä enemmän tietoa. Kaikki laitteet eivät siis välttämättä ole tähän menetelmään tarpeeksi nopeita. Tekstuurin resoluutiota voi kuitenkin pienentää tarpeen mukaan, mutta se vaikuttaa pelin mielekkyyteen. Totesin, että tekstuurin voi lähettää 60 %:n laadulla ilman että vastaanotettu näkymä kärsii liikaa (tämä toki riippuu laitteesta, isommalla näytöllä ero on huomattavampi). Pikseleitä on vastaavasti vähemmän, jolloin kuva on paljon sulavampi vaikka laskentateho ei olisi niin korkea. Sinänsä käsittelyn hitaus on ihan ymmärrettävää, koska väritiedot sijoittuvat lopulta kuitenkin näytönohjaimen muistiin. Huomattavaa on kuitenkin testilaitteiden rajallinen laskentateho. Tekstuurin resoluution ja laadun määrittely tapahtuu manuaalisesti Unity-editorissa. Kaikki toiminnallisuus tapahtuu TextureDataSender-komponentissa (listaus 4).

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

public class TextureDataSender : MonoBehaviour
{
    public Camera renderCamera;

    // These are the dimensions of the target device
    // and they specify the size of the texture
```

```
    public int TargetWidth = 1440;
    public int TargetHeight = 600;

    // The texture size is scaled by this quality factor in the beginning.
    // The lower it is, the faster the texture is loaded on client.
    [Range(0.1f, 1.0f)]  public float Quality = 1.0f;

    private Texture2D target;
    private RenderTexture renderTexture;

    private NetworkHost networkHost;

    void Start()
    {
        int x = Convert.ToInt32(TargetWidth * Quality);
        int y = Convert.ToInt32(TargetHeight * Quality);

        target = new Texture2D(x, y, TextureFormat.RGB24, false);
        renderTexture = new RenderTexture(x, y, 24);

    #if (UNITY_STANDALONE)
        networkHost = GetComponent<NetworkHost>();
    #endif

        GameObject targetObj = GameObject.Find("Target");

         if (targetObj) {
             ResizePlaneToScreen(targetObj);
         }
    }

    void ResizePlaneToScreen(GameObject plane)
    {
        float height = Camera.main.orthographicSize * 2.0f;
        float width = height * Screen.width / Screen.height;
        plane.transform.localScale = new Vector3(width, 0.1f, height) / 10.0f;

        material = plane.renderer.material;
    }

#if (UNITY_STANDALONE)

    Dictionary<int, bool> playerBuffer;

    void OnPostRender()
    {
        if (!(Network.isServer && Network.connections.Length > 0))
            return;

       // Pixels are only sent when the client has received them.
       // Therefore sending is disabled until notified later.
       playerBuffer = new Dictionary<int,bool>(networkHost.PlayerStatus);
       foreach (KeyValuePair<int, bool> player in playerBuffer)
       {
           if (player.Value != false)
               continue;

                foreach (NetworkPlayer networkPlayer in Network.connections)
                {
                    if (player.Key == int.Parse(networkPlayer.ToString()))
                    {
                        SendCameraPixels(networkPlayer);
```

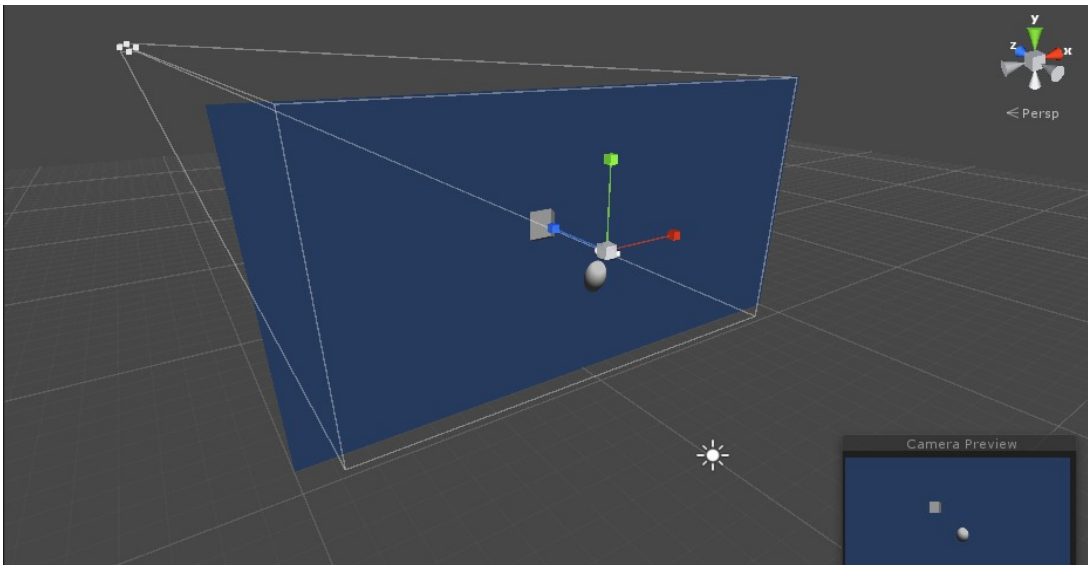
```
                        networkHost.PlayerStatus[player.Key] = true;
                    }
                }
            }
        }

    void SendCameraPixels(NetworkPlayer player)
    {
        // Camera is first rendered to a texture, then the renderTexture
        // must be set active for it to work with the ReadPixels function.
        renderCamera.targetTexture = renderTexture;
        renderCamera.Render();
        RenderTexture.active = renderTexture;

        target.ReadPixels(new Rect(0, 0, TargetWidth, TargetHeight), 0, 0);
        target.Apply();

        RenderTexture.active = null;

        networkView.RPC("SendTexture", player, target.EncodeToJPG());
    }
#endif

    Material material = null;

    [RPC]
    void SendTexture(byte[] pixelData)
    {
        target.LoadImage(pixelData);
        material.SetTexture(0, target);

        // A message is sent to the server when the client has received
        // and processed the texture, so that it can send a new one.
        networkView.RPC("TextureReceived", RPCMode.Server, Network.player,
                        false);
    }

    [RPC]
    void TextureReceived(NetworkPlayer player, bool status)
    {
        networkHost.PlayerStatus[int.Parse(player.ToString())] = status;
    }
}
```

Listaus 4. Menetelmä streamingin toteuttamiseksi. Kameran sisältö lähetetään palvelimelta asiakkaalle.

Tekstuurin koon asettelu

Luvussa 6.5 totesin, että tekstuuri täytyy saada mobiililaitteen näytön kokoiseksi. Ensimmäinen askel on PC:ltä lähetettävän tekstuurin koon asettaminen. Sen ei ole välttämätöntä olla täysin samankokoinen laitteen kanssa, mutta kuvasuhteen on oltava sama. Jos mobiililaitteen resoluutio on esimerkiksi 1280 x 800, tekstuurin resoluution täytyy siis olla samassa suhteessa (16:10). Muuten kuva voi näyttää litistyneeltä. Unity-editorissa on mahdollista muuttaa lähetettävän tekstuurin kokoa.

Toinen askel on tekstuurin asettaminen tasoon. Se on geometrinen, tasainen objekti. Sen tulisi täyttää koko mobiililaitteen näyttö, joten sen kokoa täytyy muuttaa automaattisesti resoluution perusteella. Asiakkaan kohdalla käytetään ortografista projektiota eli kamera ei käytä perspektiiviä (kuva 27). Silloin syvyysnäkymää ei ole. Sitä ei tarvita pelkän kaksiulotteisen kuvan näyttämiseen. Tason koko lasketaan ResizePlaneToScreen-funktiossa (listaus 4).



Kuva 27. Tekstuurin skaalaus tasoon (Unity-editorin näkymä).

7.8 Asymmetrisyys tilasynkronisaatiota hyödyntäen

Tässä osiossa käsitellään asymmetrisyyden toteuttamista pelitilojen synkronaatiota hyödyntäen: Mitä toiminnallisuuksia asymmetriseen peliin täytyy tässä tapauksessa toteuttaa? Miten ja mitkä tiedot lähetetään eri laitteiden välillä?

Jotta asymmetrisyys voidaan toteuttaa tilasynkronisaation avulla, tietyt toiminnallisuudet on toteutettava peliin. Kirjoitin aiemmin automaattisesta järjestelmästä, jolla on mahdollista lähettää pelin IP-osoite lähiverkkoon PC:ltä sekä etsiä sitä mobiililaitteella. Silloin liittyminen onnistuu automaattisesti, eikä IP-osoitetta tarvitse syöttää erikseen. Mikä on seuraava askel? Mielekästä olisi, että peliin liittyvien nimet olisivat näkyvissä ennen pelin alkamista.

Monessa verkkopelissä on toteutettu jonkinlainen aula, jossa pelaajat odottavat pelin alkamista. Tämän jälkeen peli käynnistetään jokaisella laitteella – kaikilla laitteilla ladataan siis uusi taso. Itse pelinäkymässä taas täytyy huolehtia siitä, että jokainen saa käyttöön oman pelaajahahmon. Vihollisten ja ammusten tulisi ilmestyä jokaisen pelaajan ruudulle, ja niiden täytyisi myös tuhoutua. Kun pelaaja liittyy mukaan kesken pelin, tulisi pelimaailman tila olla sama kuin muillakin. Tilasynkronisaation perusta on kuvattu luvussa 6.4 – käytännössä pyritään hyödyntämään paljon etäproseduurikutsuja (RPC). Projektin toteutustavaksi oli määritelty tilasynkronisaatio, joten tässä osiossa esimerkkinä käytetään toteutettua prototyyppiä.

Pelaajan tiedot ja peliaula

Peliaulan toteuttamiseksi palvelin tarvitsee jokaisen pelaajan tiedot. Vähintään tarvitaan pelaajan nimi, jotta se voidaan näyttää aulassa. Nimi syötetään pelissä kenttään (kuva 28), ja laitteesta riippuen joko palvelin käynnistetään (PC) tai aletaan etsimään peliä (mobiili). Nämä tiedot on tarpeellista säilyttää palvelimella, sillä niitä voidaan tarpeen tullen hyödyntää myöhemmin. Esimerkiksi Unityn NetworkPlayer-tietorakenne sisältää tietoja liittyneestä pelaajasta, muun muassa IP-osoitteen.

Kuva 28. Pelaajan nimen syöttäminen ja palvelimen aloitus.

Tiedot voidaan lähettää palvelimelle etäproseduurikutsulla, kun pelaaja liittyy peliin. Prototyypissä on muodostettu PlayerData-rakenne, joka muutetaan tieto- ja lähetettäessä binääritaulukoksi C#-kielen kirjastoja hyödyntäen (listaus 5). Etäproseduurikutsut eivät hyväksy esimerkiksi viittauksien lähettämistä, koska ne voivat vaihdella laitteiden välillä. Jos haluaa lähettää paljon tietoa kerralla, se voidaan muuttaa binääritaulukoksi tai lähettää tiedot erillisinä parametreina ja muodostaa tietorakenne vastaanottajan puolella.

```
using UnityEngine;
using System.Collections;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;

[System.Serializable]
public class PlayerData
{
    public string Name;

    [System.NonSerialized]
    public NetworkPlayer NetworkPlayer;

    public PlayerData(string name)
    {
        this.Name = name;
    }
}

// Source: http://stackoverflow.com/questions/1446547/how-to-convert-an-object-
to-a-byte-array-in-c-sharp
public static class Converter
{
    public static byte[] SerializeToByteArray(this object obj)
    {
        if (obj == null)
        {
            return null;
        }
        var bf = new BinaryFormatter();
        using (var ms = new MemoryStream())
        {
            bf.Serialize(ms, obj);
            return ms.ToArray();
        }
    }

    public static T Deserialize<T>(this byte[] byteArray) where T : class
    {
        if (byteArray == null)
        {
            return null;
        }
        using (var memStream = new MemoryStream())
        {
            var binForm = new BinaryFormatter();
            memStream.Write(byteArray, 0, byteArray.Length);
```

```
            memStream.Seek(0, SeekOrigin.Begin);
            var obj = (T)binForm.Deserialize(memStream);
            return obj;
        }
    }
}
```

Listaus 5. Pelaajan tietojen rakenne ja binääritaulukoksi muuntaminen.

Palvelimella tiedot tallennetaan omaan GameData-rakenteeseen (listaus 6). Kun asiakas liittyy peliin, se lähettää siis palvelimelle etäproseduurikutsun, jossa on pelaajan syöttämät tiedot. NetworkPlayer-rakenne täytyy lähettää erikseen, koska sitä ei voida muuttaa binäärimuotoon.

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

public class GameData
{
    public List<PlayerData> Players { get; private set; }

    public GameData()
    {
        Players = new List<PlayerData>();
    }

    public void AddPlayer(PlayerData player)
    {
        if (Network.isServer) {
            Players.Add(player);
        }
    }

    public void RemovePlayer(NetworkPlayer networkPlayer)
    {
        PlayerData player = Players.Find((PlayerData data) =>
                                         data.NetworkPlayer == networkPlayer);

        if (player != null) {
            Debug.Log("Player removed: " + player.Name);
            Players.Remove(player);
        }
    }

    public void Reset()
    {
        Players.Clear();
    }
}
```

Listaus 6. Rakenne pelitiedoille pelaajien tietojen tallettamista varten.

Asiakkaan verkkotoiminnallisuus on toteutettu NetworkConnection-luokkaan (listaus 7), kun taas palvelimella on NetworkHost-luokka (listaus 1). Kumpikin niistä hallitsee omaa osa-aluettaan peliin liittymisestä. Tässä tapauksessa NetworkConnection suorittaa tietojen lähettämisen, kun yhteys palvelimeen on muodostettu OnConnectedToServer-funktiossa. Peliin liittyminen tapahtuu aiemmin kuvatun IPReceiver-luokan avulla, jonka toiminnallisuus aloitetaan, kun pelaaja alkaa etsiä peliä.

```
using UnityEngine;
using System.Collections;

public class NetworkConnection : MonoBehaviour
{
    public bool IsConnected { get; private set; }

#if ((UNITY_ANDROID || UNITY_IPHONE || UNITY_WP8))

    public int remotePort = 1025;
    private IPReceiver _IPReceiver;
    private GameDataCommunication dataCommunication;
    public string PlayerName { get; set; }

    void Start()
    {
        _IPReceiver = GetComponent<IPReceiver>();
        dataCommunication = GetComponent<GameDataCommunication>();
    }

    public void SearchGame()
    {
        _IPReceiver.StartReceivingIP();
    }

    public void ConnectToServer(string IP)
    {
         Network.Connect(IP, remotePort);
    }

    public void Disconnect()
    {
        Network.Disconnect();
        ClearConnection();
    }

    void ClearConnection()
    {
        _IPReceiver.ClearIP();
        Network.RemoveRPCsInGroup(0);
        PlayerName = null;
        IsConnected = false;
    }


    #region --- Client network events ---
```

```
    void OnConnectedToServer()
    {
        IsConnected = true;
        dataCommunication.SendPlayerData(new PlayerData(PlayerName),
                                         Network.player);
    }

    void OnDisconnectedFromServer(NetworkDisconnection info)
    {
        ClearConnection();
        Destroy(GameObject.Find("NetworkManager"));
        Application.LoadLevel(0);
    }

    #endregion
#endif
}
```

Listaus 7. Asiakasyhteyden toiminta.

Pelitietojen lähettäminen tapahtuu GameDataCommunication-luokkaa hyödyntäen (listaus 8). Varsin tyypillinen tapa lähettää tietoa on määritellä etäproseduurikutsu, joka ottaa vastaan lähetettävät tiedot parametrina (ks. ReceivePlayerData-funktio listauksessa 8). Tämä kutsu lähetetään palvelimelle, jolloin se vastaanottaa ja tallentaa tiedot. Kun tiedot on lähetetty onnistuneesti, pelaajan nimi syötetään taulukkoon, jonka kaikki peliin liittyneet näkevät (kuva 29). Myös tämä tapahtuu etäproseduurikutsulla. Vain palvelimella on mahdollisuus aloittaa peli aulasta painiketta painamalla.

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

/// <summary>
/// Central messaging component between server and client for sending game data.
/// </summary>
public class GameDataCommunication : MonoBehaviour

{
    public GameObject LabelPrefab;
    private GameData gameData;

    public bool GameStarted { get; private set; }

    void Start()
    {
        gameData = new GameData();
        GameStarted = false;

        // Without this serialization with BinaryFormatter will break on iOS.
        #if (UNITY_STANDALONE_OSX || UNITY_IPHONE)
            System.Environment.SetEnvironmentVariable("MONO_REFLECTION_SERIALI
              ZER", "yes");
```

```
        #endif
    }

    public void RemovePlayer(NetworkPlayer player)
    {
        gameData.RemovePlayer(player);
    }

    public void AddPlayer(PlayerData playerData, NetworkPlayer player)
    {
        // The NetworkPlayer must be sent seperately,
        // since it cannot be serialized to custom binary.
        // It is supported as such by Unity RPCs.
        playerData.NetworkPlayer = player;

        gameData.AddPlayer(playerData);
        Debug.Log("Added new player: " + playerData.Name +
         " (" + player.ipAddress + ")");

        if (!GameStarted)
        {
            networkView.RPC("AddPlayerLabel", RPCMode.AllBuffered,
            playerData.Name);
        }
    }

    public void SendPlayerData(PlayerData playerData,
                               NetworkPlayer networkPlayer)
    {
        byte[] data = Converter.SerializeToByteArray(playerData);
        networkView.RPC("ReceivePlayerData", RPCMode.Server, data,
                        networkPlayer);
    }

    [RPC]
    void AddPlayerLabel(string name)
    {
        GameObject label = NGUITools.AddChild(GameObject.Find("NameGrid"),
                                              LabelPrefab);
        label.GetComponent<UILabel>().text = name;

        GameObject.Find("NameGrid").GetComponent<UIGrid>().repositionNow = true;
    }

    [RPC]
    void ReceivePlayerData(byte[] player, NetworkPlayer networkPlayer)
    {
        PlayerData playerData = player.Deserialize<PlayerData>();
        AddPlayer(playerData, networkPlayer);
    }
}
```

Listaus 8. Komponentti pelitietojen välittämiseen palvelimen ja asiakkaan välillä.

Kuva 29. Peliaula (palvelimen näkymä).

7.8.1 Tasojen lataaminen verkon välityksellä

Myös tasojen lataaminen tapahtuu GameDataCommunication-luokassa. Lataamista varten siihen lisätään alempana listatut toiminnallisuudet. Lataamisessa hyödynnetään etäproseduurikutsua, jonka palvelin lähettää puskuroituna kaikille pelaajille (LoadLevel-funktio, listaus 9). Käsittelen puskurointia tarkemmin peliobjektien luomisen yhteydessä. Käytännössä puskuroitu etäproseduurikutsu tarkoittaa, että toiminto kutsutaan myös kaikille peliin liittyville pelaajille. Kutsu jää siis palvelimen muistiin. Etäproseduurikutsu lähetetään, kun PC-pelaaja painaa ”Start Game” -painiketta (listaus 10).

```
[RPC]
void LoadLevel(string name, int levelPrefix)
{
    Network.SetSendingEnabled(0, false);
    Network.isMessageQueueRunning = false;

    Network.SetLevelPrefix(levelPrefix);

    Application.LoadLevel(name);
}

void OnLevelWasLoaded(int level)
{
    Network.isMessageQueueRunning = true;
    Network.SetSendingEnabled(0, true);
```

```
        if (level > 0 && (Network.isServer || Network.isClient))
        {
            Spawner.SpawnLevel();
            GameStarted = true;
        }
    }
```

Listaus 9. Tason lataaminen verkon välityksellä.

```
    void StartGame()
    {
        if (Network.isServer)
        {
            networkHost.networkView.RPC("LoadLevel", RPCMode.AllBuffered,
                                        "Level1", Application.loadedLevel);
        }
    }
```

Listaus 10. Pelin aloittaminen ja etäproseduurikutsun lähettäminen tasojen lataamista varten.

7.8.2 Peliobjektien luominen ja tuhoaminen

Erittäin merkittävä osa pelejä on luoda pelinäkymään uusia peliobjekteja, esimerkiksi pelaaja ja vihollisia. Monissa peleissä pelaaja pystyy tuhoamaan vihollisia, jolloin ne täytyy myös poistaa pelinäkymästä. Prototyypin aiheena on ammuskelupeli, joten vihollisten tuhoaminen on isossa roolissa.

Peliobjektien luominen tapahtuu Unity-ympäristössä GameObject.Instantiate-funktion avulla. Ne voidaan asettaa haluamaansa paikkaan asetettujen koordinaattien perusteella. Mutta miten peliobjekteja voidaan luoda verkon välityksellä? Jos esimerkiksi palvelin kutsuu GameObject.Instantiate-funktiota paikallisesti, mitään ei tapahdu tietenkään asiakkaan pelissä. Tämän vuoksi Unity määrittelee erikseen Network.Instantiate-funktion, joka huolehtii peliobjektien luomisesta kaikkien pelaajien näkymiin. Se on käytännössä puskuroitu etäproseduurikutsu, jolloin peliobjektit luodaan myös myöhemmin peliin liittyville pelaajille.

Tavallisessa pelissä peliobjektit voidaan tuhota GameObject.Destroy-funktiolla. Vastaavasti Network.Destroy on määritelty niiden tuhoamiseksi kaikista peleistä. Suuri ero Network.Instantiate-funktioon on se, että objektien tuhoaminen verkossa ei ole puskuroitu etäproseduurikutsu. Se aiheuttaa potentiaalisen ongel-

man: jos vihollinen on jo tuhottu, miten voidaan estää sen ilmestyminen muihin peleihin? Unity mahdollistaa näiden puskuroitujen kutsujen poistamisen Network.RemoveRPCs-funktion avulla, joka huomioi myös Network.Instantiate-kutsut.

Tietenkin on myös mahdollista määritellä kokonaan oma järjestelmä peliobjektien luomiseksi sekä tuhoamiseksi, jolloin paikallisia GameObject.Instantiate ja GameObject.Destroy -funktioita kutsutaan etäproseduurikutsujen kautta. Tämä kuitenkin aiheuttaa lisää ongelmia, sillä pelaajan täytyy huolehtia siitä, että networkView-komponenttien tunnukset (ID) sekä omistajuus täsmäävät. Tällä tavalla voidaan kuitenkin tehdä paljon varmempi järjestelmä, koska sitä voi itse hallita täysin.

Prototyypissä hyödynnetään pääasiassa Network.Instantiate- ja Network.Destroy -funktioita. Esimerkiksi ampuminen on toteutettu tällä tavalla. Jotta se olisi mahdollisimman tarkkaa, palvelin hallitsee ammusten luomisen, tuhoamisen sekä liikuttamisen. Pelinkehittäjä voi tietenkin valita, miten paljon autoritaarisuutta käytetään. Yksinkertaistettu Gun-luokka havainnollistaa ammusten luomista (listaus 11). Jos jotakin peliobjektin ominaisuutta halutaan muokata sen luomisen jälkeen verkossa, täytyy lähettää erikseen etäproseduurikutsu, jotta se tulee voimaan kaikille pelaajille. Tätä on hyödynnetty ammusten suunnan ja vahingon asettamisessa. Tuhoaminen tapahtuu vastaavasti BulletBehaviour-komponentin (listaus 12) DestroyAndClean-funktiossa, jolloin tuhotaan siihen liittyvä puskurin sisältö. Tässä on myös havainnollistettu palvelimen autoritaarisuus: liikkuminen ja törmäysten määrittely tapahtuu palvelimella, tiedot lähetetään erikseen asiakkaille.

```
using UnityEngine;
using System.Collections;

public class Gun : MonoBehaviour
{
    public ParticleSystem fireEffect;
    public GameObject bulletPrefab;
    private Animator animator;

    void Start()
    {
        animator = GetComponent<Animator>();
    }
```

```
    public void FireBullet()
    {
        GameObject bullet = null;

        if (!animator.GetBool("IsShooting"))
        {
            animator.SetBool("IsShooting", true);

            if (!fireEffect.isPlaying) {
                fireEffect.Play();
            }

            if (Network.isServer)
            {
                Vector3? direction = CalculatePerspectiveDirection();

                if (direction.HasValue)
                {
                   bullet = Network.Instantiate(bulletPrefab, transform.position,
                                                transform.rotation, 0)
                                                as GameObject;

                    if (direction != Vector3.zero)
                    {
                        bullet.networkView.RPC("SetDirection", RPCMode.All,
                                               direction.Value);
                    }
                }
            }
        }
    }

    [RPC]
    public void FireBulletTarget(Vector3 targetPos)
    {
        GameObject bullet = null;

        if (!animator.GetBool("IsShooting"))
        {
           animator.SetBool("IsShooting", true);

           if (!fireEffect.isPlaying) {
               fireEffect.Play();
           }

           if (Network.isServer)
           {
              bullet = Network.Instantiate(bulletPrefab, transform.position,
                                           transform.rotation, 0) as GameObject;

              Vector3 direction = (targetPos - transform.position).normalized;
              bullet.networkView.RPC("SetDirection", RPCMode.All, direction);

              int baseDamage = bullet.GetComponent<BulletBehaviour>().BaseDamage;
              bullet.networkView.RPC("SetDamage", RPCMode.All, baseDamage / 2);
           }
        }
    }

    /// <summary>
    /// Animation event: called when shooting animation has done playing.
    /// </summary>
```

```
    void ExitShootAnim()
    {
        animator.SetBool("IsShooting", false);
        fireEffect.Stop();
    }
}
```

Listaus 11. Peliobjektien luominen verkon välityksellä (aseen ammukset).

```
using UnityEngine;
using System.Collections;

public class BulletBehaviour : MonoBehaviour
{
    public int BaseDamage = 50;
    public float LifeTime = 2.0f;

    private Vector3 direction;
    private float speed = 200.0f;
    private float raycastRange = 5.0f;

    private bool hasHit = false;

    [RPC]
    public void SetDirection(Vector3 dir)
    {
        direction = dir;

        Ray ray = new Ray(transform.position, direction);
        RaycastHit hit;

        if (Physics.Raycast(ray, out hit, raycastRange))
        {
            if ((hit.collider.gameObject.tag == "Target" ||
                hit.collider.gameObject.tag == "Obstacle") && !hasHit)
            {
                StartCoroutine("OnHitObject", hit.collider.gameObject);
            }
        }
    }

    [RPC]
    public void SetDamage(int damage)
    {
        BaseDamage = damage;
    }

    void FixedUpdate()
    {
        if (!Network.isClient)
        {
            transform.position += direction * speed * Time.deltaTime;
            CheckForCollisions();

            LifeTime -= Time.deltaTime;

            if (LifeTime <= 0) {
                DestroyAndClean();
            }
        }
    }
```

```
void CheckForCollisions()
{
    if (hasHit)
        return;

    RaycastHit hit;
    Ray ray = new Ray(transform.position, direction);

    if (Physics.Raycast(ray, out hit, raycastRange))
    {
        if ((hit.collider.gameObject.tag == "Target" ||
            hit.collider.gameObject.tag == "Obstacle") && !hasHit) {
            StartCoroutine("OnHitObject", hit.collider.gameObject);
        }
    }
}

IEnumerator OnHitObject(GameObject obj)
{
    if (!hasHit)
    {
        hasHit = true;

        // We wait until the end of the next frame, so the trail
        // displays on close shots as well.
        for (int i = 0; i < 2; i++)
        {
            yield return new WaitForEndOfFrame();
        }

        if (gameObject != null)
        {
            CombatStatus status = obj.GetComponentInChildren<CombatStatus>();

            if (status && !status.IsDead && Network.isServer)
            {
                status.networkView.RPC("Hit", RPCMode.All, BaseDamage);
            }
            else if (status && !status.IsDead && !Network.isClient)
            {
                status.Hit(BaseDamage);
            }
        }

        DestroyAndClean();
    }
}

/// <summary>
/// Destroy bullet and clear any RPCS related to it.
/// </summary>
void DestroyAndClean()
{
    if (Network.isServer)
    {
        Network.RemoveRPCs(this.networkView.viewID);
        Network.Destroy(this.gameObject);
    }
    else if (!Network.isClient && !Network.isServer)
    {
        Destroy(this.gameObject);
```

```
        }
    }
}
```

Listaus 12. Ammuksien hallinta palvelimella sekä peliobjektien tuhoaminen verkon välityksellä.

7.8.3 Peliin liittyminen sen alkamisen jälkeen

Kuten kävi ilmi, luodakseen peliobjektit peleihin täytyy käyttää Network.Instantiate-funktiota tai omaa järjestelmää. Mutta entä sitten, kun pelinäkymä ladataan? Unity-editorissa on mahdollista luoda tasoja graafisesti. Tässä peliprototyypissä tarkoitus oli asettaa Unity-editorissa vihollisten pesät tasoihin tällä tapaa. Kun pelaaja liittyy mukaan peliin – myös tuhotut objektit ladataan. Tässä on järkeä, sillä Network.Destroy ei ole puskuroitu funktio, eikä Network.Instantiate-funktiota ole käytetty, jolloin peliobjektin luomista ei millään tapaa poisteta.

Jälleen kerran olisi mahdollista hyödyntää omaa järjestelmää, mutta tässä prototyypissä hyödynnetään Network.Instantiate-funktiota. Tämän ratkaisemiseksi ainakin kaksi mahdollisuutta: 1) peliobjektit poistetaan alussa näkymästä, ja palvelin luo ne Network.Instantiate-funktiolla tai 2) näkymään asetetaan peliobjekteja, jotka merkitsevät pelkästään siihen luotavan peliobjektin sekä sijainnin (listaus 13). Tämä mahdollistaa sen, että taso ladataan myös myöhemmin liittyville pelaajille oikeassa tilassa. Ympäristö on kuitenkin mahdollista ladata tavallisesti, esimerkiksi seinät eivät tuhoudu pelin aikana.

```
public static void SpawnLevel()
 {
     #if (UNITY_STANDALONE)

             GameObject[] spawnerLocations =
             GameObject.FindGameObjectsWithTag("SpawnEnemy");

             foreach (GameObject spawner in spawnerLocations)
             {
                 if (Network.isServer)
                 {
                     Network.Instantiate(Resources.Load("Prefabs/EnemyNest"),
                                         spawner.transform.position,
                                         spawner.transform.rotation, 0);
                 }
             }
```

```
            GameObject fpsStart =
            GameObject.FindGameObjectWithTag("SpawnFPS");
            Spawner.SpawnFPSPlayer(fpsStart.transform.position);

    #endif

    #if ((UNITY_ANDROID || UNITY_IPHONE || UNITY_WP8))
            GameObject tpsStart =
            GameObject.FindGameObjectWithTag("SpawnTPS");
            Spawner.SpawnTPSPlayer(tpsStart.transform.position);
    #endif
}
```

Listaus 13. Pelikentän peliobjektien luominen niin, että ne ladataan ja tuhotaan myös liittyville pelaajille.

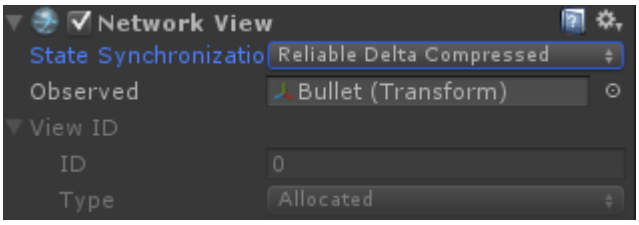
7.8.4 Liikkumisen synkronointi ja tilasynkronisaatio

Koska asymmetrinen peli toteutettiin lähiverkon välityksellä, pelaajan liikettä ennakoivilla menetelmillä ei ole paljon merkitystä. Tämä johtuu siitä, että latenssia laitteiden välillä on vähän. On kuitenkin mahdollista, että langattomassa verkossa on häiriösignaaleja tai mobiililaitteen laitteisto voi vaikuttaa yhteyteen. Oikeassa tuotannossa mahdolliset häiriötekijät sekä pelin hidastuminen täytyy käsitellä, jolloin ainakin pelaajan liike täytyy varmistaa sulavaksi joka tilanteessa.

Pelinkehityksen aikana kävi ilmi, että eniten liikkeen sujuvuuteen vaikuttaa itse pelin nopeus: jos esimerkiksi mobiililaite toimii erittäin hitaasti, silloin myös pelaaja liikkuu kaikissa laitteissa ”tökkien”. Tämä johtuu siitä, että prototyypissä asiakas lähettää sijaintitiedot. Jos liikkumisesta halutaan mahdollisimman sulavaa tilanteessa kuin tilanteessa, siinä tulisi hyödyntää OnSerializeNetworkView-funktiota sekä interpolaatiota, jolloin pelaaja asetettaisiin liikkumaan myös sijaintitietojen väliset arvot. Esimerkiksi jos pelaaja saisi paketit (1, 0, 0) sekä (2, 0, 0), hahmo liikkuisi välillä (1..2, 0, 0) sen sijaan, että se hyppäisi suoraan uuteen koordinaattiin. Prototyypissä käytetään Unityn sisäänrakennettua tilasynkronisaatiota liikkeen ja sijaintitietojen yhtenäistämiseksi, sillä se on prototyypin tarpeisiin ja laadulliseen tasoon riittävä.

Liikkumisen toteuttaminen lähiverkossa toimivassa asymmetrisessä pelissä on joka tapauksessa huomattavasti yksinkertaisempaa kuin Internetin välityksellä toimivassa verkkopelissä. Unity-editorissa tilasynkronisaation menetelmä tarvit-

see asettaa vain networkView-komponentille (kuva 30). Täytyy kuitenkin huomioida, että lähetysnopeuden täytyy olla korkeampi kuin tavanomaisessa verkkopelissä. Se ei kuitenkaan haittaa, koska kaistaa on lähiverkossa käytettävissä paljon. Prototyypissä peliobjektit luodaan pääasiassa palvelimella, joten sijaintitietojen lähetys tapahtuu sulavasti. Yksi suurempi poikkeus on mobiilipelaajan hahmo.



Kuva 30. NetworkView-komponentin asetukset (paikkatietojen tilasynkronisaatio).

Omien tietojen lähettäminen tilasynkronisaatiolla

Jatkuvasti päivitettävien ja käytettävien tietojen lähettämiseen on parempi käyttää etäproseduurikutsujen sijasta OnSerializeNetworkView-funktiota. Etäproseduurikutsut eivät ole tähän käyttötarkoitukseen yhtä luotettavia, sillä ne eivät esimerkiksi seuraa tietojen muuttumista. Se toimii samalla periaatteella kuin sijaintien seuraaminen, mutta ohjelmoimalla voidaan määritellä, mitä tietoja lähetetään. Esimerkiksi tässä prototyypissä sitä käytettiin vihollisten nopeustietojen lähettämiseen palvelimelta asiakkaalle (listaus 14). Sen avulla voitaisiin toteuttaa esimerkiksi interpolaatio liikkumiseen tai pelaajien elinvoiman seuranta.

```
void OnSerializeNetworkView(BitStream stream, NetworkMessageInfo info)
{
    // Agent velocity is sent to the client using OnSerializeNetworkView.
    // This is because server handles the free roaming of enemies.
    // Walking animations would not be played otherwise (velocity remains at 0)

    Vector3 velocity = Vector3.zero;

    if (networkView.isMine && stream.isWriting)
    {
        velocity = agent.velocity;
        stream.Serialize(ref velocity);
    }
    else
    {
        stream.Serialize(ref velocity);
        agent.velocity = velocity;
    }
```

```
}
```

Listaus 14. Reaaliaikaisen tiedon lähettäminen OnSerializeNetworkViewin avulla.

7.8.5 Yhteenveto ja muita huomioita

Tilansynkronisaatiota ja etäproseduurikutsuja käytettäessä ei ole yhtä oikeaa vastausta asymmetrisyyden toteuttamiseksi. Yleisellä tasolla on mahdollista pohtia pelejä yhdistäviä elementtejä, mutta toiminnallisuuden toteuttaminen on aina pelikohtaista. Pelinkehittäjä voi itse valita, miten paljon palvelin käsittelee käskyjä ja tietoja (autoritaarisuus). Erityisesti hyödynnetään etäproseduurikutsuja, joilla voidaan kutsua toiminnallisuuksia halutuista kohteista. Sijaintitiedot voidaan synkronoida Unityn sisäänrakennetulla järjestelmällä, mutta parempi liikkuvuus vaatii vähintään interpolaatiota. Tilasynkronisaatiota voidaan hyödyntää OnSerializeNetworkView-funktion avulla myös muihin tietoihin.

Erityisen tärkeää on huomioida yhteyden muodostaminen, peliobjektien luominen ja tuhoaminen, pelaajan liittyminen tai lähteminen kesken pelin (jolloin pelinäkymän tulisi olla sama) sekä tarpeellisen pelitiedon lähettäminen palvelimen ja asiakkaiden välillä.

Merkittävää pelitilojen synkronoimisessa on sen aiheuttama ylimääräinen työmäärä. Toiminnallisuudet eivät toimi täysin samalla tavalla kuin verkottomassa pelissä. Tämä vaatii esimerkiksi etäproseduurikutsujen suunnittelemista siten, että viittauksia muihin objekteihin ei käytetä. Prototyyppiä kehittäessä huomasin, että etäproseduurikutsujen ja tilapäivitysten poistaminen ei ole niin yksinkertaista kuin oletin: ilman tarkkoja varmistuksia ei voi aina olla täysin varma siitä, lähetetäänkö jokin kutsu tai tilapäivitys peliobjektin tuhoamisen välissä. Jos peliin halutaan yksinpeli, tapaukset täytyy huomioida ohjelmakoodissa. Yksi mahdollisuus on myös aloittaa paikallinen verkkopeli, johon muut eivät voi liittyä.

Prototyypin kehittämisen perusteella totean, että pelitilojen synkronointi toimisi erittäin hyvin esimerkiksi ylimääräisen käyttöliittymän synkronisoimiseksi pelin

kanssa tai tablet-tietokoneen käyttämiseksi ohjaimena, sillä se vaatii varsin vähän toiminnallisuuksia. Asymmetrisyys on joka tapauksessa mahdollista toteuttaa pelitilojen synkronointia hyödyntämällä kokonaiseen peliin, mutta se vaatii enemmän työtä.

8 Tulokset

Tämän työn aikana kehitettiin peliprototyyppi, joka toteuttaa asymmetrisen pelikokemuksen PC:n ja mobiililaitteiden välille. Käytännössä tämä tapahtui niin, että pelaajilla on toisistaan poikkeavat näkymät (kuvat 31 ja 32), jotka vaikuttavat myös pelaajan rooleihin. Mobiililaitteita käyttävät pelaajat ovat heikompia avustajia, kun taas PC-pelaajalla on enemmän ampumisvoimaa. Tämä toteutettiin synkronoimalla paikalliset pelitilat.

Tämän työn tärkein asia oli selvittää, miten asymmetrinen peli voidaan toteuttaa PC:n ja mobiililaitteiden välille. Lisäksi tutkimuskysymyksin kuuluivat: Miten hyvin asymmetrisyys soveltuu PC-alustalle? Miten mobiililaitteita voidaan hyödyntää PC-pelaamisessa?

Kuva 31. PC-pelaajan näkymä pelistä.

Kuva 32. Mobiilipelaajan näkymä pelistä.

Miten asymmetrinen peli toteutetaan teknisesti?

Asymmetrisyys toteutetaan PC:n ja mobiililaitteiden välille ympäristössä, jossa on mahdollista kehittää pelejä monelle laitteelle yhtäaikaisesti. Se vaatii kaksi eri versiota pelistä: yhdessä on palvelimen ominaisuudet, toinen on suunniteltu peliin liittymistä varten. Itse asymmetrisen pelikokemuksen toteuttaminen vaatii verkko-ohjelmoinnin hyödyntämistä. PC:ltä voidaan lähettää mobiililaitteeseen kuva streamingiä hyödyntäen tai pelitilat voidaan synkronoida lähettämällä tietoja tavallisen verkko-ohjelmoinnin tapaan. Unityn käyttö ja verkko-ohjelmointi olivat merkittävässä roolissa asymmetrisen prototyypin toteutuksessa myös Speckin tutkimuksessa (2013), joka hyödynsi Photon-verkkoympäristöä.

Miten hyvin asymmetrisyys soveltuu PC-alustalle?

Pelitilojen synkronoinnilla toteutettuna yhteys oli prototyypissä toimiva, ja se tukee helposti montaa pelaajaa kerralla. Menetelmä täytti asymmetrisyydelle asetetut vaatimukset. Asymmetrisyyden toteuttamiseksi hyödynnettiin laitteiden välisiä eroja: PC:llä ohjataan hahmoa ensimmäisessä persoonassa näppäimistön ja hiiren avulla, kun taas mobiililaitteella ohjaaminen tapahtuu kosketuksen avulla. Pelin syötteet ja hallinta täytyy siis ottaa huomioon, jos ylimääräistä laitetta aiotaan hyödyntää.

Prototyyppi osoittaa, että moninpelin luominen PC:n ja mobiililaitteen välille on mahdollista Unity-ympäristössä. Asymmetrisyys kuuluu läheisesti pelisuunnitteluun ja vaikuttaa siihen, mitä kukin pelaaja tekee. Sen kannattavuus ja mielekkyys on pääasiassa pelikohtaista.

Streaming osoittautui epävakaammaksi, sillä se vaatii mobiililaitteelta enemmän laskentatehoa kuin itse pelin suorittaminen. Kun testikuva lähetettiin low-end -laitteeseen, se toimi hitaammin kuin itse prototyyppi. Jos asymmetrisyyden haluaisi toteuttaa streamingiä hyödyntäen, täytyisi sen toimivuutta tutkia tarkemmin useilla eri laitteilla ja oikeassa pelitilanteessa. Unity-ympäristössä se toteutettiin RenderTexture-ominaisuuden avulla sekä pikselitietojen lähettämisellä JPG-muodossa palvelimelta asiakkaalle.

Taustatutkimuksessa selvisi, että nykyiset konsolit hyödyntävät kannettavia laitteita. Esimerkiksi on mahdollista pelata konsolipelejä käsikonsolin avulla. Wii U:n GamePadin avulla tai Xboxin SmartGlass teknologialla on myös mahdollista näyttää esimerkiksi ylimääräisiä tietoja ruudulla tai toteuttaa asymmetrinen moninpeli. Yhteys on toteutettu näissäkin tapauksissa lähiverkon avulla. Myös muutamat PC-pelit ovat hyödyntäneet mobiililaitteita, joko apusovelluksen muodossa, tarjoamalla laitteiden välisen asymmetrisen pelikokemuksen tai yksinkertaisesti toteuttamalla asymmetriset roolit ilman lisälaitteita.

Miten mobiililaitteita voidaan hyödyntää PC-pelaamisessa?
Mobiililaitteen käyttötapoja selvitettiin Wii U -konsolin pelejä analysoimalla: Sitä voidaan käyttää ylimääräisenä ohjaimena hyödyntämällä esimerkiksi kosketusnäytön ja kiihtyvyysantureiden ominaisuuksia. Toisella ruudulla voidaan näyttää tietoja pelistä, esittää toisenlainen näkymä pelistä tai sitä voidaan käyttää ylimääräisenä pelialueena. Toisella ruudulla on mahdollista pelata pääpelistä irrallaan olevia minipelejä, jotka vaikuttavat jollakin tavalla takaisin.

Joitakin laitteistoratkaisuja on kehitetty PC:n ja mobiililaitteiden väliseen hallintaan: OnLive sekä Nvidia Gaming Portable tarjoavat jopa mahdollisuuden pelata PC-pelejä mobiililaitteen avulla. Joissakin näppäimistöissä on mahdollisuus näyttää ylimääräisiä tietoja peleistä.

9 Pohdinta

Lopuksi on tarpeellista tarkastella asymmetrisyyttä itsessään. Onko se kannattavaa? Mitä hyviä ja huonoja puolia siinä on? Miten asymmetrisyyttä voitaisiin hyödyntää jatkossa? Tämän vuoksi asymmetrisyydelle on asetettava perusteluja ja kritiikkiä tutkimuksen pohjalta.

9.1 Perusteluja ja kritiikkiä asymmetrisille peleille

Tutkimuksen aikana tulin siihen lopputulokseen, että asymmetrisyyden täytyy tuoda pelille huomattavasti lisäarvoa, että se on kannattavaa. Tämä perustuu etenkin vertailuun New Super Mario Bros. U:sta sekä Rayman Legendsistä. Jos toinen pelaaja ei koe rooliaan merkittäväksi tai miellyttäväksi, on asymmetrisyys pelissä silloin turhaa. Tätä voi puntaroida kysymällä, miten hauska peli on yksin pelatessa ja asymmetrisenä moninpelinä. Hauskuus on toki elementtinä vaikea ennustaa, mutta esimerkiksi prototyypin tekemällä se selviää hyvin. Toisaalta voidaan miettiä tavanomaista moninpeliä: millä tavalla asymmetrisyys on parempi vaihtoehto?

Ylimääräisen laitteen lisääminen tarkoittaa pelaajan kohdalla jonkinlaista vaivannäköä. Kokeeko pelaaja esimerkiksi karttatoiminnon tarpeeksi hyödylliseksi, että hän ottaa sovelluksen käyttöön mobiililaitteessa? Erityisesti pienissä ominaisuuksissa tai apusovelluksissa tämä voi osoittautua liian suureksi kynnykseksi. Periaatteessa asymmetriset roolit tai ominaisuudet voidaan toteuttaa myös ilman ylimääräisiä laitteita, joten niitä ei välttämättä edes tarvita. Kynnys pelinkehittäjän puolesta voi olla verkko-ohjelmointiin ja monelle laitteelle kehittämiseen menevä aika. Onneksi Unityn kaltaiset monialustaiset pelimoottorit ja kehitysympäristöt helpottavat prosessia huomattavasti.

Joskus ylimääräinen laite voi kuitenkin olla pelillisesti erittäin merkittävä. Esimerkiksi jaetussa näytössä on omat ongelmansa, pelaajat voivat muun muassa nähdä toisensa. Kosketusnäyttöä sekä mobiililaitteen muita ominaisuuksia voi-

daan hyödyntää monella tavalla, kuten neljännessä luvussa esitetyistä esimerkeistä kävi ilmi. Esimerkiksi ohjaaminen laitetta kallistelemalla on täysin erilainen kokemus tavalliseen hiireen ja näppäimistöön verrattuna.

Asymmetrisyyden suurin etu on sen avaamissa mahdollisuuksissa. Sen täyttä potentiaalia ei ole todennäköisesti vielä näytetty, sillä Wii U -konsoli on ollut olemassa vasta kaksi vuotta. Ilmiö voisi olla tunnetumpi, jos useampi peli hyödyntäisi sitä. Ylimääräisten laitteiden käyttäminen sekä erilaisten roolien määrittäminen on todella avara tila pelisuunnittelulle. Asymmetrisyyttä hyödyntämällä voitaisiin kehittää uudenlaisia moninpelikokemuksia.

Asymmetrisyys on ehdottomasti toimiva myös PC-alustalla. Tämä johtuu siitä, että mobiililaitteet ovat sopivia sen toteuttamiseen. Niissä ei useimmiten tosin ole painikkeita kuten Wii U:n GamePadissa. Mobiililaitteet käyttävät nykyisin kapasitiivista kosketusnäyttöä ja tukevat useita kosketuksia kerralla. GamePadin kosketusnäyttö on resistiivinen ja tukee vain yhtä kosketusta (Bivens 2012), mikä voidaan kokea rajoittavaksi tekijäksi. Lisäksi nuorella ikäpolvella mobiililaitteet ovat hyvin laajassa käytössä – ylimääräistä lisälaitetta ei siis pelaamista varten tarvitsisi. Tablet-tietokoneella pelikokemus on kuitenkin mahdollisesti miellyttävämpi.

Alussa esitettyjen tilastojen perusteella tablet-tietokoneiden määrä on huomattavassa kasvussa. Siinä mielessä mobiililaitteita hyödyntäville PC-peleille voisi olla asiakaskuntaa. Joissain tapauksissa voidaan käyttää ylimääräisiä hallintalaitteita kuten Oculus Riftiä. Ne tuovat lisäarvoa asymmetrisyyden toteuttamiseksi PC:llä. Silloin laitteen yleisyys voi muodostua ongelmaksi, mutta se voidaan välttää mahdollistamalla pelaaminen myös ilman laitetta. Speckin (2013, 68) mukaan laitteiden välisellä pelaamisella on tosin arvoa pelinkehityksessä.

Yksi huomattavasti asymmetrisyyttä ja mobiililaitteita hyödyntävistä yrityksistä on Ubisoft. Yritys on kehittänyt pelejä Wii U -konsolille sekä hyödyntänyt mobiililaitteita esimerkiksi Assassin's Creed IV-, The Division- ja The Watch Dogs -peleissä. Ubisoft on todennut, että apusovellusten idea on se, että pelaajat voivat vuorovaikuttaa pelin kanssa merkittävällä tavalla myös peliajan ulkopuolella.

He myös huomasivat, että toisen näytön käyttäminen yhtäaikaisesti vaatii liikaa huomiota pelaajalta. Ubisoftin Chris Earlyn mukaan pelien apusovelluksia käytetään todennäköisesti vielä 5 – 10 vuotta. (Maiberg 2014.) Koska ihmiset eivät voi pelata peliä kahdella näytöllä aktiivisesti, se on huomattava rajoitus peli-ideoille. Asymmetrisissä moninpeleissä tätä ongelmaa ei tosin esiinny, koska pelaajat keskittyvät omaan rooliinsa ja laitteeseensa.

9.2 Pelitilojen synkronoinnin ja streamingin vertailu

Tässä työssä esitettiin kaksi merkittävää tapaa toteuttaa asymmetrinen peli PC:n ja mobiililaitteiden välillä. Ne kuvattiin kuudennessa luvussa, jossa käsiteltiin asymmetrisyyden toteuttamista. Paikallisesti suoritettujen pelien synkronointi hyödyntää verkko-ohjelmointia siten, että pelien välillä lähetetään pelkästään tietoa. Se muistuttaa hyvin läheisesti tavanomaista verkkopelin kehitystä. Streaming on erityisesti konsolien hyödyntävä menetelmä, jossa tietty osa pelistä voidaan lähettää kuvana mobiililaitteeseen. Lisäksi asymmetristä peliä varten tarvitaan pelipalvelin sekä kaksi eri versiota pelistä, joissa on tarvittavat ominaisuudet.

Kuvan lähettäminen streamingiä hyödyntäen on erittäin hyvä tapa toteuttaa asymmetrisyys siksi, että ylimääräistä verkko-ohjelmointia ei juurikaan tarvita. Mobiililaitteen tarvitsee vain vastaanottaa kuva sekä lähettää näppäinpainallukset palvelimelle. Kaikki peliin liittyvä laskenta tapahtuu palvelimella. Se voi tosin rajoittaa jossain määrin mahdollisten pelaajien määrää. Kuvan lähettäminen mahdollistaa huomattavasti paremman graafisen laadun. Toisaalta taas kuvan käsittely mobiililaitteessa voi viedä huomattavasti laskentaehoa. Tässä työssä streamingiä ei käytetty oikeassa pelitilanteessa, joten sen toimivuus käytännössä on kyseenalainen. Kuten aiemmin mainittiin, konsolit hyödyntävät omaa teknologiaa sekä laitteistoa streamingin toteuttamiseksi. PC:llä yksittäiselle pelinkehittäjälle tämä on ongelma, koska yhteistä teknologiaa ei varsinaisesti ole. Toteutettu menetelmä kuvan lähettämiseksi ei ole todennäköisesti ideaalinen; esimerkiksi videolla voitaisiin saada parempi lopputulos.

Pelitilojen synkronointi on siinä mielessä varmempi tapa, että teknologia on varsin yksinkertaista ja suoraviivaista. Lähetettävän tiedon määrä on huomattavasti pienempi. Siinä hyödynnetään tavanomaista verkko-ohjelmointia lähiverkossa. Tätä tapaa käytettiin prototyypin toteuttamisessa, ja se todettiin toimivaksi. Toisaalta todettiin, että verkko-ohjelmointi vaatii tavalliseen peliohjelmointiin verrattuna enemmän aikaa. Synkronisaation toteuttaminen on myös pelikohtaisempaa streamingiin verrattuna. Lähiverkon hyödyntäminen tosin helpottaa asymmetrisyyden toteuttamista.

9.3 Ideoita ja huomioita jatkokehitystä varten

Tämän tutkimuksen ulkopuolelle jäi muutama asia, joita voitaisiin tutkia tarkemmin. Työssä keskityttiin toteuttamaan asymmetrinen peli käyttäen Unity 4:n verkkokirjastoja, jotka perustuvat RakNetiin. Olisi mielenkiintoista selvittää, helpottaako esimerkiksi ulkoisten kirjastojen käyttäminen asymmetrisyyden kehittämistä. Tätä varten on olemassa valmiita ratkaisuja, jotka rakentuvat Unityn ympärille. Esimerkiksi Photon ja uLink ovat tällaisia kirjastoja. Lisäksi Unityn verkkokirjastot uusiutuvat Unity 5:n kehityssyklin aikana (Unity Technologies 2014i), joten lähestymistapa asymmetrisyyden toteuttamiseen saattaa jossakin määrin muuttua. Vaikka Unityn verkkokirjastot ovat jo varsin helppokäyttöiset, kehittyneempi ratkaisu voisi nopeuttaa kehitystä, jolloin asymmetrisyyden toteuttamiseen kuluisi vähemmän aikaa.

Työssä ei käytetty muita lisälaitteita kuin älypuhelinta sekä tablet-tietokonetta. Olisi mielenkiintoista nähdä, miten asymmetrisyys toimii muiden lisälaitteiden kuten Oculus Riftin yhteydessä. Prototyypissä mobiililaitteita käytettiin varsin yksipuolisesti, käytännössä hyödynnettiin kosketusnäytön ominaisuuksia hahmon hallinnassa. Kuten kävi ilmi luvusta 4.6, sitä voitaisiin käyttää monella muulla tapaa. Esimerkiksi pelihahmoa voitaisiin hallita laitetta kallistamalla, mobiililaitteella voitaisiin tähdätä PC:n näyttöä kohti tai käyttää mikrofonia ja kameraa jonkin toisen peli-idean yhteydessä. Myös Speck totesi tutkimuksessaan (2013, 63 – 66), että laitteistoja voitaisiin käyttää monipuolisemmin.

Lopuksi, streamingiä voitaisiin tutkia vielä tarkemmin. Olisiko esimerkiksi mahdollista toteuttaa kuvan lähettäminen video-enkooderia hyödyntäen? Periaatteessa pelikuvan lähettäminen JPG-muotoisena ei ole ideaali ratkaisu. Sitä voitaisiin myös kokeilla hyödyntää oikeassa pelitilanteessa, jolloin menetelmän todellinen potentiaali selviäisi.

Tässä opinnäytetyössä esitellyn teoreettisen ja käytännön tiedon perusteella olisi mahdollista toteuttaa ainakin erilaisia peli-ideoita, jotka hyödyntävät asymmetrisyyttä ja mobiililaitteita PC:llä. Prototyyppejä voitaisiin tehdä siis lisää. Tässä työssä toteutettu prototyyppi oli pelillisesti varsin yksinkertainen, erilaisten ominaisuuksien toteuttaminen olisi sen vuoksi tärkeää. Pelialalle tulokset merkitsevät erityisesti uusia innovaatioita sekä eri laitteiden monipuolista hyödyntämistä ja integraatiota, jopa potentiaalia erilaisten pelien tuotantoon.

Joka tapauksessa asymmetrisyys tarjoaa paljon mahdollisuuksia kehittää uudenlaisia pelejä. PC:llä mobiililaitteita ei vielä käytetä kovin paljon; on kyseenalaistettavaa tullaanko niitä käyttämään enemmän peleissä myöhemmin. Erityisesti näen asymmetrisyyden potentiaalin yhteistoiminnallisissa peleissä, vaikka ylimääräisiä laitteita ei hyödynnettäisikään.

Lähteet

Abrams, L. 2004. TCP and UDP Ports Explained. http://www.bleepingcomputer.com/tutorials/tcp-and-udp-ports-explained. 24.9.2014.

Abrego, C. 2013. Rayman Legends Wii U - (2048p) Co Op - Part 1. http://www.youtube.com/watch?v=wtucXRFGU3M. 13.10.2014.

AfterDawn. 2014. NAT. http://fin.afterdawn.com/sanasto/selitys.cfm/nat. 24.9.2014.

AnandTech. 2013. Capsule Keyboard Roundup: Entries from Logitech, SteelSeries, and Corsair. http://www.anandtech.com/show/7198/capsule-keyboard-roundup-entries-from-logitech-steelseries-and-corsair. 18.11.2014.

Appburst. 2014. PlanetSide 2 Mobile Uplink App. http://www.appburst.com/apps/planetside-2-mobile-uplink-app. 21.10.2014.

ARIN. 2014. IPv4 and IPv6. https://www.arin.net/knowledge/ipv4_ipv6.pdf. 26.9.2014.

Bivens, D. 2012. Nintendo Explains Choice for Single-Touch Wii U GamePad Functionality. http://www.nintendoworldreport.com/news/30748/nintendo-explains-choice-for-single-touch-wii-u-gamepad-functionality. 16.11.2014.

Blizzard Entertainment. 2014. World of Warcraft Mobile Armory. http://eu.battle.net/wow/en/shop/mobile-armory/. 21.10.2014.

BNGR. 2012. Nintendo Land Review. http://www.youtube.com/watch?v=SYsWtBRYvr4. 15.10.2014.

CGNRundertow. 2012. Game Boy Advance link cable video game hardware review. http://www.youtube.com/watch?v=kKtaqP5Gygo. 10.10.2014.

Children's Technology Review. 2013. The Legend of Zelda Windwaker. http://www.youtube.com/watch?v=c90Yd3v4yaw. 15.10.2014.

CNET. 2014. Trying out PS4 remote play on the Sony Xperia Z3v. http://www.youtube.com/watch?v=Ar5RrBpCG5Y. 10.10.2014.

CNN. 2001. Nintendo unveils Gamecube launch plans. http://edition.cnn.com/2001/TECH/fun.games/08/24/gamecube.release.idg/ 10.10.2014.

Crecente, B. 2012. The surprising (mundane) tech behind the Wii U's magical GamePad. www.polygon.com/2012/11/16/3653294/wii-u-range-test-gamepad. 26.9.2014.

Davidson, J. 2013a. http://www.technobuffalo.com/wp-content/uploads/2013/06/Game-and-Wario-2.jpg. 18.11.2014.

Davidson, J. 2013b. http://www.technobuffalo.com/wp-content/uploads/2013/06/Game-and-Wario-4.jpg. 18.11.2014.

Ebay. 2014. Can I Connect My Nintendo DS to My Nintendo Wii? http://www.ebay.com/gds/Can-I-Connect-My-Nintendo-DS-to-My-Nintendo-Wii-/10000000177628528/g.html. 10.10.2014.

Edge. 2013. Still Playing: Nintendo Land. http://media.edge-online.com/wp-content/uploads/edgeonline/2012/12/Nintendo-Land-Luigis-Ghost-Mansion.jpg. 18.11.2014.

Edward. 2005. Sony PSP. http://commons.wikimedia.org/wiki/File:Sony_PSP.png. 18.11.2014.

ESA. 2013. Essential facts about the computer and video game industry. http://theesa.com/facts/pdfs/ESA_EF_2013.pdf, 9.10.2014.

ESA. 2014. The Evolution of Mobile Games. http://www.theesa.com/games-improving-what-matters/mobile-games.asp. 9.10.2014.

Evan-Amos. 2011. A black Nintendo Game Boy Advance, a handheld video game system. http://commons.wikimedia.org/wiki/File:Game-Boy-Advance-Blk.jpg. 18.11.2014.

ExtremeTech. 2014. OnLive resurrected, allows you to stream your Steam games to any device. http://www.extremetech.com/gaming/177981-onlive- resurrected-allows-you-to-stream-your-steam-games-to-any-device. 20.10.2014.

Farrington, J. 2012. Wii U Preview: Just What Is Asymmetrical Gameplay? http://dorkshelf.com/2012/07/11/wii-u-preview-just-what-is-asymmetrical-gameplay/. 26.9.2014.

Faultline Games. 2014. http://faultlinegames.com/games/ns2combat/. 21.10.2014.

Final Fantasy Wiki. 2014. Chocobo World. http://finalfantasy.wikia.com/wiki/Chocobo_World. 3.10.2014.

Logitech 2014. G19S Gaming Keyboard. http://gaming.logitech.com/fi-fi/product/g19s-lcd-gaming-keyboard. 21.10.2014.

GameFAQs. 2014. PlayStation 3. http://www.gamefaqs.com/ps3/927750-playstation-3/data. 10.10.2014.

GameSpot. 2013. Worldwide industry sales to reach $93 billion in 2013 – Report. http://www.gamespot.com/articles/worldwide-industry-sales-to-reach-93-billion-in-2013-report/1100-6415832/. 9.10.2014.

Giant Bomb. 2014. Asymmetric Gameplay. www.giantbomb.com/asymmetric-gameplay/3015-7566/. 25.10.2014.

Groenendijk, F. 2012. New Super Mario Bros U Cheats. http://86bb71d19d3bcb79effc-d9e6924a0395cb1b5b9f03b7640d26eb.r91.cf1.rackcdn.com/wp-content/uploads/2012/11/new-super-mario-bros-u-cheats.jpg. 18.11.2014.

GT News. 2014. WATCH DOGS ctOS GAMEPLAY DEMO. http://www.youtube.com/watch?v=IVWAZUGZFjU. 20.10.2014.

Honorof, M. Logitech K480 Types on Every Device. http://www.tomsguide.com/us/logitech-k480-types-device,news-19449.html. 21.10.2014.

IANA. 2014. Service Name and Transport Protocol Port Number Registry. http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml. 26.9.2014.

IDC. 2014. Smartphone OS Market Share, Q2 2014. http://www.idc.com/prodserv/smartphone-os-market-share.jsp. 9.10.2014.

IGN CA. 2000. Game Boy Advance: It's finally unveiled. http://ca.ign.com/articles/2000/08/24/game-boy-advance-its-finally-unveiled. 10.10.2014.

IGN. 2013. Xbox One's SmartGlass Functionality Explained. http://www.ign.com/articles/2013/09/06/xbox-ones-smartglass-functionality-explained. 10.10.2014.

IGN. 2014a. Duck Hunt Cheats & Codes. http://www.ign.com/cheats/games/duck-hunt-nes-7158. 25.10.2014.

IGN. 2014b. GamePad features – Mario Kart 8. http://www.ign.com/wikis/mario-kart-8/GamePad_Features. 15.10.2014.

IGN. 2014c. Companion App (iFruit). http://www.ign.com/wikis/gta-5/Companion_App_(iFruit). 21.10.2014.

Jason Rybka. 2014. Co-op. http://vgstrategies.about.com/od/strategyglossary/g/coop.htm. 10.10.2014.
Kainy 2014. What's Kainy? http://www.kainy.com/whatiskainy.html. 20.10.2014.
Kotaku. 2014. 6 Minutes of Evolve. http://www.youtube.com/watch?v=_wmklRY2JXI. 21.10.2014.
Leap Motion. 2014. Leap Motion. https://www.leapmotion.com/product. 21.10.2014.
NetMarketShare. 2014. Mobile/Tablet Operating System Market Share. http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1&qpsp=180&qpnp=9&qptimeframe=M. 1.11.2014.
Newzoo. 2014. Infographic: The Global Mobile Landscape Reloaded. http://www.newzoo.com/infographics/infographic-global-mobile-landscape-reloaded/. 9.10.2014.
Nintendo. 2014a. Connecting the Nintendo GameCube Game Boy Advance Link Cable. http://www.nintendo.com/consumer/systems/nintendogamecube/gcn_agb_connect.jsp. 10.10.2014.
Nintendo. 2014b. Game Boy Player – Using a Game Boy Advance as a Controller. http://www.nintendo.com/consumer/systems/nintendogamecube/gameboyplayer_agb_controller.jsp. 10.10.2014.
Nintendo. 2014c. Nintendo Buyer's Guide. http://www.nintendo.com/consumer/buyers_guide.jsp. 10.10.2014
Nintendo. 2014d. Wii U technical specs. http://www.nintendo.com/wiiu/features/tech-specs/. 9.10.2014.
Nintendo. 2014e. http://gameandwario.nintendo.com/_ui/img/minigames/taxi/screen1.png. 18.11.2014.
Nintendo Life. 2013. http://www.nintendolife.com/reviews/wiiu/rayman_legends. 15.10.2014.
Nintendo Master. 2014. http://www.nintendo-master.com/fichiers/2014/2/19/1392808309.jpg. 18.11.2014.
Nvidia 2014a. The Ultimate Portable Gaming Family. http://shield.nvidia.com/. 21.10.2014.
Nvidia 2014b. PC Gaming Made Portable. http://shield.nvidia.com/play-pc-games/. 21.10.2014.
Nvidia. 2014c. The Ultimate Gaming Portable. http://shield.nvidia.com/portable-game-player/. 21.10.2014.
Plant, A. 2012. Wii U: Could Asymmetric Multiplayer Work for Zelda? http://www.gengame.net/wp-content/uploads/2012/11/zelda-asymmetric-battle-quest.jpg. 18.11.2014.
Maiberg, E. 2014. Ubisoft and the evolution of second screen gaming. http://killscreendaily.com/articles/ubisoft-and-evolution-second-screen-gaming/. 16.11.2014.
MobyGames. 2008. Pac-Man Vs. (GameCube). http://www.mobygames.com/game/gamecube/pac-man-vs/screenshots/gameShotId,279478/. 18.11.2014.
Moby Games. 2014a. New Super Mario Bros. U. http://www.mobygames.com/game/wii-u/new-super-mario-bros-u. 13.10.2014.
Moby Games. 2014b. New Super Luigi U. http://www.mobygames.com/game/wii-u/new-super-luigi-u. 13.10.2014.
Moby Games. 2014c. Game & Wario. http://www.mobygames.com/game/wii-u/game-wario. 13.10.2014.

Moby Games. 2014d. Nintendo Land Wii U Review. http://www.mobygames.com/game/wii-u/nintendo-land/reviews/reviewerId,179990/. 15.10.2014.
Moby Games. 2014e. Mario Kart 8. http://www.mobygames.com/game/wii-u/mario-kart-8. 15.10.2014.
Moby Games. 2014f. Legend of Zelda: The Wind Waker. http://www.mobygames.com/game/wii-u/legend-of-zelda-the-wind-waker. 15.10.2014.
Moby Games. 2014g. Frozen Synapse. http://www.mobygames.com/game/windows/frozen-synapse. 21.10.2014.
Mode 7. 2013. Frozen Synapse iPad Cross-Play Demo. http://www.youtube.com/watch?v=MgBBXr9VafI. 21.10.2014.
Oculus VR. 2014. Oculus Rift – Virtual Reality Headset for Immersive 3D Gaming. http://www.oculus.com/rift/. 21.10.2014.
OnLive. 2014a. OnLive Desktop:Real PC Apps on iPad and Android. http://desktop.onlive.com/overview. 20.10.2014.
OnLive. 2014b. About OnLive. https://games.onlive.com/about. 20.10.2014.
OnLive. 2014c. Platforms. https://games.onlive.com/about_device. 20.10.2014.
OnLive. 2014d. OnLive Games Bundle. https://games.onlive.com/games#selectGamebundle. 20.10.2014.
OnLive. 2014e. https://lh6.ggpht.com/0vf_SjRmkDkmSvbKzeVVdZ_-JxQmW7fArnde1A7kHQjRfniJYzfbuPUrDWz5fOtfU2A=h900. 18.11.2014.
PCMag. 2014. Definition of: Smartphone. http://www.pcmag.com/encyclopedia/term/51537/smartphone. 17.11.2014.
Pope, T. 2014. How to use Xbox One SmartGlass. http://www.gottabemobile.com/2013/12/07/use-xbox-one-smartglass/. 10.10.2014.
RakNet. 2014a. C# and Unity. http://www.jenkinssoftware.com/raknet/manual/csharpunity.html. 26.9.2014.
RakNet. 2014b. FAQ. http://www.raknet.net/raknet/manual/faq.html. 25.9.2014.
Speck, R. 2013. Designing Asymmetrical Collaborative Gameplay for Heterogeneous Device Ecosystems. https://idea.library.drexel.edu/islandora/object/idea%3A4256/datastream/OBJ/download/Designing_Asymmetrical_Collaborative_Gameplay_for_Heterogeneous_Device_Ecosystems.pdf. 24.11.2014.
Roccat. 2014a. What is POWER-GRID? http://power-grid.roccat.org/en/Home/. 20.10.2014.
Roccat. 2014b. Download. http://power-grid.roccat.org/en/Download/. 20.10.2014.
Roccat. 2014c. FUTURE READY. http://www.future-ready.roccat.org/. 21.10.2014.
Rouse, M. 2005. UDP (User Datagram Protocol). http://searchsoa.techtarget.com/definition/UDP 24.9.2014.
Rouse, M. 2006. Local area network (LAN). http://searchnetworking.techtarget.com/definition/local-area-network-LAN. 26.9.2014.
Rouse, M. 2008. TCP/IP (Transmission Control Protocol/Internet Protocol). http://searchnetworking.techtarget.com/definition/TCP-IP. 24.9.2014.
Rouse, M., Harschutz, J. 2005. Sockets. http://whatis.techtarget.com/definition/sockets. 25.9.2014.
Rouse, M. 2010. Wireless LAN (WLAN or Wireless Local Area Network).

http://searchmobilecomputing.techtarget.com/definition/wireless-LAN. 26.9.2014.
Rouse, M., Blair, E. 2014. Latency. http://whatis.techtarget.com/definition/latency. 26.9.2014.
Sage, S. 2013. Tom Clancy's The Division to let mobile players get in on post-apocalyptic console action. http://www.imore.com/tom-clancys-division-let-mobile-players-get-post-apocalyptic-console-action. 18.11.2014.
Snyder, J. 2014. Asymmetrical Gameplay: Gimmick or Revolution? http://www.theoryofgaming.com/asymmetrical-gameplay-gimmick-or-revolution/. 18.11.2014.
Sony Computer Entertainment America. 2014. PS Vita and PS3 Remote Play Guide. https://support.us.playstation.com/app/answers/detail/a_id/3815/~/ps-vita-and-ps3-remote-play-guide.10.10.2014.
Splechta, M. 2014. Watch Dogs ctOS App Preview: Bring some hacking action into the real world. http://download.gamezone.com/uploads/image/data/1163285/Watch_Dogs_ctOS_-_Mobile_CompanionApp_TrafficLight_Tablet_Collage.png. 18.11.2014.
Statista. 2014. Statistics and facts on Mobile Gaming. http://www.statista.com/topics/1906/mobile-gaming/. 9.10.2014.
Takimata. 2012. Wii U Console and Gamepad transparent background. http://commons.wikimedia.org/wiki/File:Wii_U_Console_and_Gamepad.png. 18.11.2014.
Technopedia. 2014. Mobile Device. http://www.techopedia.com/definition/23586/mobile-device. 17.11.2014.
TechPowerUp. 2009. New Logitech G-series Peripherals Unveiled. http://www.techpowerup.com/81001/new-logitech-g-series-peripherals-unveiled.html. 26.10.2014.
TechPowerUp. 2014. ROCCAT Launches Nyth MMO Mouse and Skeletr Keyboard. http://www.techpowerup.com/203807/roccat-launches-nyth-mmo-mouse-and-skeltr-keyboard.html. 21.10.2014.
TechRadar. 2014. Sony: no to bringing PS4 remote play to other phones – ex cept our own. http://www.techradar.com/news/phone-and-communications/mobile-phones/sony-no-to-bringing-ps4-remote-play-to-other-phones-except-our-own-1264334. 26.10.2014.
TechRadar. 2013. Microsoft Xbox SmartGlass: what you need to know. http://www.techradar.com/news/gaming/microsoft-xbox-smartglass-what-you-need-to-know-1084819/1. 10.10.2014.
TheBitBlock. 2013a. New Super Luigi U (GamePad Co-Op). http://www.youtube.com/watch?v=4i-RXYGozB0. 13.10.2014.
TheBitBlock. 2013b. THE FOLD - Game & Wario (Taxi + KungFu). http://www.youtube.com/watch?v=eAUUFUWKQAw. 15.10.2014.
TheBitBlock. 2013c. THE FOLD - Game & Wario (Fruit + Ski). http://www.youtube.com/watch?v=xeRkly5iTFw. 15.10.2014.
TheBitBlock. 2013d. THE FOLD - Game & Wario (Arrow + Patchwork). http://www.youtube.com/watch?v=ddFVgPXr_Rg. 15.10.2014.
TheBitBlock. 2013e. THE FOLD - Game & Wario (Islands + Gamer). http://www.youtube.com/watch?v=4NXab5cM5fE. 15.10.2014.
TheSixthAxis. 2013. Tablet Controlled Drones And Tom Clancy's The Division. http://www.thesixthaxis.com/2013/09/05/tablet-controlled-drones-and-tom-clancys-the-division/. 21.10.2014.
Tilastokeskus. 2013. Yli neljännes 75–89-vuotiaista käyttää internetiä. http://www.stat.fi/til/sutivi/2013/sutivi_2013_2013-11-07_tie_001_fi.html.

9.10.2014.
Tilastokeskus. 2011. Tieto- ja viestitätekniikan käyttö 2011. http://www.stat.fi/til/sutivi/2011/sutivi_2011_2011-11-02_fi.pdf. 9.10.2014.
Titus, A. 2009. How the Internet Works in 5 Minutes. http://www.youtube.com/watch?v=7_LPdttKXPc. 26.9.2014.
Unbox Therapy. 2011. PlayStation Vita Remote Play Demo. http://www.youtube.com/watch?v=kEhiiZf6v24. 10.10.2014.
Unity Answers. 2013. Networking – MasterServer. http://answers.unity3d.com/questions/571052/networking-masterserver.html. 24.9.2014.
Unity Answers. 2011. Streaming textures possible? http://answers.unity3d.com/questions/142751/streaming-textures-possible.html. 24.9.2014.
Unity Technologies. 2014a. Effortlessly unleash your game on the world's hot test platforms. http://unity3d.com/unity/multiplatform. 26.9.2014.
Unity Technologies. 2014b. High Level Networking Concepts. http://docs.unity3d.com/Manual/net-HighLevelOverview.html. 24.9.2014.
Unity Technologies. 2014c. Platform Dependent Compilation. http://docs.unity3d.com/Manual/PlatformDependentCompilation.html. 23.9.2014.
Unity Technologies. 2014d. Network Reference Guide. http://docs.unity3d.com/Manual/NetworkReferenceGuide.html. 24.9.2014.
Unity Technologies. 2014e. Master Server. http://docs.unity3d.com/Manual/net-MasterServer.html. 26.9.2014.
Unity Technologies. 2014f. State Synchronization Details. http://docs.unity3d.com/Manual/net-StateSynchronization.html. 25.9.2014.
Unity Technologies. 2014g. Network Views. http://docs.unity3d.com/Manual/net-NetworkView.html. 24.9.2014.
Unity Technologies. 2014h. Render Texture. http://docs.unity3d.com/Manual/class-RenderTexture.html. 26.9.2014.
Unity Technologies. 2014i. Announcing UNET – New Unity Multiplayer Technology. http://blogs.unity3d.com/2014/05/12/announcing-unet-new-unity-multiplayer-technology. 16.11.2014.
VG24/7. 2014. You're not getting dual-GamePad gaming on Wii U anytime soon. http://www.vg247.com/2014/06/19/dual-gamepad-wii-u-games-not-on-the-cards-right-now/. 9.10.2014.
Webopedia. 2014. MTU - Maximum Transmission Unit. http://www.webopedia.com/TERM/M/MTU.html. 3.10.2014.
What Is Playstation 4. 2014. PS4 remote play. http://whatisplaystation4.com/playstation-4-specs-and-capabilities/ps4-remote-play/. 10.10.2014.
WittzGaming. 2012. Wittz Co-Op Play: New Super Mario Bros U - Part 1. http://www.youtube.com/watch?v=0v94dU1vkwA. 26.10.2014.
Zeldapedia. 2014. Tingle Tuner. http://zelda.wikia.com/wiki/Tingle_Tuner. 10.10.2014.
Zheng, L. 2012. Hands-on with Xbox SmartGlass: Xbox Remote, Internet Explorer, Forza Horizon, Dance Central 3. http://www.istartedsomething.com/wp-content/uploads/2012/11/smartglass3.jpg. 18.11.2014.

Peliprototyypin käyttö ja lataus

Projektin tulokset on ladattavissa osoitteesta:
https://www.dropbox.com/sh/ozlkf2hkm7ge017/AAAdTBgWVHSUOyS9Pjhu4ORYa?dl=0

Lataaminen

Jos haluat kokeilla vain itse peliä, lataa pelkästään kansio nimeltä **Game Prototype**. Kokeellinen tekstuurin lähetys löytyy kansiosta **TextureStream**, kun taas projektin lähdekoodit on kansiossa **Scripts**. Paina Dropboxin ”download” painiketta haluamassasi kansiossa.

Käyttäminen

Lähdekoodia voidaan kopioida, käyttää ja muokata mihin tahansa tarkoituksiin vapaasti. Binääritiedostot ovat vain omaan, ei-kaupalliseen käyttöön. Tarvitaan kaksi tiedostoa, .apk Androidille sekä .exe Windowsille. Niitä käytetään kummallakin alustalla pelin suorittamiseksi.

Portit ja yhteys

Ohjelma käyttää portteja 1025 sekä 1026, joten niiden pitäisi olla vapaana pelin toimimiseksi. Laitteiden täytyy myös olla samassa lähiverkossa – nopeaa yhteyttä suositellaan. Varmista myös, että ohjelmat ovat palomuurin sallitulla listalla.

Tekstuurin lähetys

Sisältää kuvan lähettämisen PC:ltä mobiililaitteeseen (kameran sisältö). Kokeellinen toiminto, siinä ei ole pelielementtejä. Resoluutiota ei voida muuttaa editorin ulkopuolella, vakiona se on 1024 x 600 (tämä ei välttämättä ole yhteensopiva kaikkien laitteiden kanssa).

Peli

Toiminnallisuuksia voidaan testata yksinpelissä PC:llä, yhteyttä laitteiden välillä ei välttämättä tarvita. Aloittaaksesi palvelimen, paina ”Host Game”. Asiakkaat voivat tämän jälkeen liittyä peliin, jolloin heidän nimensä ilmestyvät ruudulle. Huomioi, että mobiilipelaajien liikkeen sulavuus riippuu mobiililaitteen nopeudesta. Peliä ei ole optimoitu, joten se ei välttämättä toimi hyvin vanhoilla laitteilla. Jos ongelmia ilmenee, pelit kannattaa käynnistää uudelleen.