OULUN AMMATTIKORKEAKOULU

Heikki Halmetoja

UNITY-PELIALUSTAN MONINPELIOMINAISUUDET

UNITY-PELIALUSTAN MONINPELIOMINAISUUDET

Heikki Halmetoja
Opinnäytetyö
Kevät 2014
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, langattomien laitteiden suuntautumisvaihtoehto

---

Tekijä(t): Heikki Halmetoja
Opinnäytetyön nimi: Unity-pelialustan moninpeliominaisuudet
Työn ohjaaja(t): Teemu Lankila, Riitta Rontu
Työn valmistumislukukausi ja -vuosi: Kevät 2014
Sivumäärä: 60 + 1 liite

Opinnäytetyön tavoitteena oli tutkia ja kokeilla Unity-pelinkehitysalustan moninpeliominaisuuksia. Työ jaettiin kahteen erilliseen osaan. Näistä ensimmäisessä kehitettiin Unityä käyttäen pelisovellus, joka mahdollistaa usean päätelaitteen välisen moninpelaamisen ilman erillistä palvelinta. Toisessa vaiheessa tehtiin selvitystyö Unitylle työn tekohetkellä saatavilla olevista moninpelattavien virtuaalimaailmojen luomiseen soveltuvista palvelinratkaisuista.

Ensimmäisen vaiheen toteutuksessa käytettiin pääasiallisena työkaluna toimineen Unityn lisäksi Microsoftin Visual Studio 2012:ta sekä Googlen Android-kehitystyökaluja. Ohjelmakoodia kirjoitettiin C#- ja Java-ohjelmointikielillä.

Toisessa vaiheessa selvitystyötä tehtiin internetiä ja selvitystyön kohteeksi valittujen tuotteiden dokumentaatiota lähteinä käyttäen. Tuotteisiin suoraan liittyvän materiaalin lisäksi tutkittiin esimerkiksi tuotteita koskevaa verkkokeskustelua ja tuotteita käyttäviä sovelluksia.

Työn tuloksena syntyi peliohjelma, joka mahdollistaa yksinkertaisen pallopelin pelaamisen Android- ja PC-laitteilla. Selvitystyön pääasialliset tulokset on kirjattu tämän opinnäytetyön liitteistä löytyvään taulukkoon sekä raportin viidenteen lukuun, joka käsittelee selvitystyötä ja tutkittuja tuotteita.

Asiasanat:
moninpelit, virtuaalimaailmat, palvelimet, Unity3D

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Wireless Electronics

Author(s): Heikki Halmetoja
Title of thesis: Multiplayer features of Unity game development platform
Supervisor(s): Teemu Lankila, Riitta Rontu
Term and year when the thesis was submitted: Spring 2014
Pages: 60 + 1 appendix

The main subject of this thesis work is Unity game development platform and its multiplayer features. The work itself was divided into two independent stages. In the first stage the goal was to develop a multiplayer gaming application. The most important requirement was that using the application to play multiplayer games should not require a dedicated server.

The aim of the second stage was to research Unity compatible multiplayer server solutions currently available on the market. Focus was on such solutions that enable the creation of persistent virtual worlds.

Main tools used in the first stage were Unity, Visual Studio 2012 and Android SDK. Programming was done in C# and Java. Information for the second stage was acquired using the internet and the documentation of the solutions selected for comparison.

Keywords:
multiplayer, virtual worlds, server software, Unity3D

ALKULAUSE

Tämä opinnäytetyö käsittelee moninpelien toteuttamista Unity3D-ympäristössä. Videopelit ja erityisesti niiden taustalla toimiva teknologia ovat olleet minulle erityinen kiinnostuksen kohde niin kauan kuin muistan, ja tämä työ tarjosi mahdollisuuden päivittää tietojani ja laajentaa ymmärrystäni näistä asioista.

Haluan kiittää Teemu Lankilaa ja Juha Väisästä mahdollisuudesta tehdä tämä opinnäytetyö Fantastec Oy:lle sekä kaikkia yrityksen työntekijöitä hyvistä hetkistä syksyn mittaan. Erityiset kiitokset esitän puolisolleni, jonka tuki ja tsemppaus on ollut korvaamattoman arvokasta koko opinnäytetyöprojektin ajan.

Oulussa 5.2.2013

Heikki Halmetoja

SISÄLLYS

1 JOHDANTO

Ihmiset ovat pelanneet pelejä jo tuhansia vuosia. Pelaamista voidaan ajatella eräänlaisena muodollisena, tiettyjen sääntöjen mukaan etenevänä leikkinä, joka mielletään yleensä sosiaaliseksi tapahtumaksi. Useimpiin tunnettuihin peleihin tarvitaankin vähintään kaksi pelaajaa.

Videopelien pelaaminen sitä vastoin on usein esitetty yksinäisenä puuhasteluna huolimatta siitä, että suuri osa varhaisimmistakin kotikäyttöön myydyistä videopeleistä oli nimenomaan usealle pelaajalle tarkoitettuja. Etupäässä internetin mahdollistama verkkomoninpelaaminen on kuitenkin muuttanut tätä asetelmaa. Verkon kautta peliseuraa on aina saatavilla ja peleihin sisäänrakennetut pikaviestimet tekevät sosiaalisten kontaktien muodostamisesta ja ylläpitämisestä vaivatonta.

Tämän opinnäytetyön aiheena on Unity-pelinkehitysalusta ja sen moninpeliominaisuudet. Moninpelaamista käsitellään sekä Unityn kontekstissa että omana kokonaisuutenaan. Verkkopelien toteuttamiseen yleensä käytettävistä teknologioista pyritään antamaan kattava kokonaiskuva. Aihepiirin laajuuden vuoksi painopiste pidetään Unityn omissa verkkopelitoiminnoissa sekä sen kanssa yhteensopivissa pelipalvelimissa.

Unity on nykyaikainen visuaalinen työkalu, joka mahdollistaa erilaisten lähdemedioiden yhdistämisen helposti ja nopeasti valmiiksi peliksi. Erilaisia objekteja, kuten grafiikkaa, ääniä ja kooditiedostoja, voidaan yhdistellä toisiinsa editorissa, joka näyttää tulokset käyttäjälle välittömästi. Työkalu on saatavilla internetistä ilmaisena versiona tietyin rajoituksin.

Opinnäytetyö jaettiin kahteen osaan. Ensimmäisen osan tavoitteena oli kehittää Unityä käyttäen peli, joka mahdollistaa moninpelaamisen usealla eri päätelaitteella verkon yli. Moninpelaamisen piti olla mahdollista ilman erillistä palvelinta. Toisessa osassa tehtiin selvitystyö markkinoilla olevista Unity-yhteensopivista palvelinratkaisuista. Erityisen kiinnostuksen kohteena olivat virtuaalimaailmojen

tai ns. MMO (Massively Multiplayer Online) -pelien luomiseen tarkoitetut palvelimet.

Työn tilaaja, Fantastec Oy, on vuonna 2009 perustettu peliyritys. Fantastec keskittyy kehittämään lapsille suunnattuja opetuksellisia tietokone- ja mobiilipelejä (1).

2 VERKKOMONINPELAAMINEN

2.1 Historiaa

Verkkomoninpelaamisen historia ulottuu ainakin 1970-luvulle. Ensimmäiset verkkoyhteyksiä hyödyntäneet pelit lienevät olleet PLATO-keskustietokone-järjestelmälle kehitettyjä. Järjestelmä koostui useista keskustietokoneista sekä hitailla verkkoyhteyksillä niihin yhteydessä olevista päätteistä, joiden kautta keskustietokoneita käytettiin. Päätteiden avulla jopa kymmenet pelaajat saattoivat osallistua samaan peliin. Tällaiset pelit olivat suosittuja lähinnä joissain yliopistoissa, jotka antoivat keskustietokoneidensa resursseja oppilaiden enemmän tai vähemmän vapaaseen käyttöön. (2.)

Kuluttajalaitteille verkon kautta pelattavia moninpelejä tarjottiin esimerkiksi BBS- eli Bulletin Board System -palveluiden kautta jo 1980-luvulla (3). Tässä vaiheessa palvelut olivat lähinnä puhelinoperaattoreiden ja harrastajien ylläpitämiä, eikä käyttäjiä ollut nykyisiin määriin verrattuna paljon. Pelit olivat tekstipohjaisia ja vuoropohjaisia, ja samanaikaisten pelaajien määrää rajoitti palvelimen yhteyksien määrä. BBS-palvelut toimivat pääasiassa puhelin-verkkojen ja analogisten modeemien avulla, ja yksi palvelimeen yhteydessä oleva käyttäjä varasi koko puhelinlinjan. Harrastustoimintana ylläpidetyillä palvelimilla oli harvoin useita puhelinlinjoja käytössään. (4.)

Vasta internetin maailmanlaajuisen yleistymisen myötä 1990-luvun puolessa välissä alkoi verkkopelaamisen harrastajien määrä toden teolla kasvaa. Vuonna 1996 julkaistu toimintapeli Quake oli yksi ensimmäisiä suurta suosiota saavuttaneita verkkopelejä. Pelissä oli useita innovatiivisia ominaisuuksia, joiden ansiosta se toimi esikuvana lukemattomille myöhemmin julkaistuille räiskintäpeleille. Lisäksi sen käyttämä palvelin-asiakastyyppinen verkkoarkkitehtuuri oli ensimmäinen laatuaan. Hyvin samankaltaisia järjestelmiä käyttävät monet pelit vielä nykyäänkin omissa verkkopelitoteutuksissaan. (5.)

Quake ei vain lisännyt verkkopelaamisen suosiota. Se teki siitä myös tavallaan demokraattisempaa. Kuka tahansa pelin hankkinut pystyi käyttämään tietokonettaan pelipalvelimena ja tarjoamaan muille mahdollisuuden pelata. Quake kasvatti myös modien eli peleihin tehtyjen lisäosien tai muokkausten suosiota. Myös modeja saattoi tehdä kuka tahansa, ja pelin julkaisijat jopa kannustivat tähän julkaisemalla pelin teknisiä tietoja ja työkaluja muokkausten tekemistä varten (6).

1990-luvulla tuloaan tekivät sittemmin valtavia pelaajamääriä keränneet MMORPG-pelit. MMORPG on englanninkielinen lyhenne ja tarkoittaa suomeksi massiivista moninpelattavaa online-roolipeliä. Näissä peleissä on pysyvä, aina avoinna oleva pelimaailma, jonne pääsee pelaamaan internetin välityksellä yleensä kuukausimaksua vastaan. Pelaajat ohjaavat hahmoja, jotka suorittavat pelimaailmassa erilaisia tehtäviä ja keräävät varusteita kehittyen samalla voimakkaammiksi. (7.) Massiivisiksi pelejä kutsutaan sen vuoksi, että samassa pelimaailmassa saattaa olla kerralla tuhansia aktiivisia pelaajia. Suosituin tämän lajityypin peleistä on vuonna 2004 julkaistu World of Warcraft, jolla oli enimmillään 12 miljoonaa aktiivista, kuukausimaksuja maksavaa käyttäjää (8).

Sittemmin erityisesti pelien muuttuminen helpommin lähestyttäviksi on tuonut uusia ihmisiä harrastuksen pariin. Ensimmäinen rahapanoksilla pelattavaa nettipokeria tarjoava palvelu aloitti toimintansa 1998, ja pelimuodon suosio on kasvanut huimasti (9). Tuoreempana ilmiönä mobiililaitteilla ja verkon sosiaa-lisissa medioissa pelattavat pelit ovat tavoittaneet suuria pelaajamääriä nopeasti. Näiden pelien myötä alan ansaintamallit ovat muuttuneet. Kiinteää maksua vastaan myytävän pelin on tietyillä markkinoilla korvannut lähes koko-naan ilmainen peli, jonka tuotto perustuu pelaajalle esitettäviin mainoksiin sekä lisäominaisuuksien myymiseen pelin sisällä (10).

2.2 Verkkopelien toteutustavat

Verkkopelit on tyypillisesti toteutettu palvelin-asiakasmallin mukaisina järjestelminä. Pelin suorittamisesta vastaa erityinen palvelinsovellus, jonka vastuulla on pelisimulaation suorittaminen. Pelaajat käyttävät pelaamiseen asiakas-

sovelluksia, jotka välittävät pelaajien syöttämät komennot palvelimelle. Palvelin prosessoi komennot, muuttaa pelin tilaa niiden mukaisesti ja lähettää tiedot muutoksista takaisin asiakkaille. (11.)

Yksinkertaista moninpelisovelluksen asiakasohjelmaa voidaan ajatella ns. tyhmänä päätteenä. Sen tehtäviin kuuluvat pelaajan syötteiden välittäminen palvelimelle, pelisimulaation tilan päivittäminen palvelimen lähettämien viestien perusteella ja sen esittäminen pelaajalle kuvana ja äänenä (11). Tällaisella rakenteella estetään pelissä huijaaminen, sillä lähettääpä asiakasohjelma millaista dataa hyvänsä, palvelin vastaa pelin sääntöjen toteuttamisesta ja pystyy estämään toiminnot, joita säännöt eivät salli (12). Käytännön toteutuk-sissa tästä perusmallista kuitenkin usein poiketaan hiukan, mistä lisää tuonnempana.

Koko pelin toteuttamiseen tarvittava palvelinjärjestelmä saattaa koostua useasta sovelluksesta tai palvelusta, jotka on joissain tapauksissa hajautettu eri laitteille. Pelisimulaatioon liittyvää laskentaa suorittavan palvelinohjelman lisäksi järjestelmään saattaa kuulua tietokantapalvelin, pikaviestipalvelin pelaajien väliseen viestintään ja WWW-palvelin peliin liittyvien tietojen esittämiseksi selaimessa. Pelien ja joukkueiden muodostaminen saattaa tapahtua erillisen matchmaking-palvelimen välityksellä.

Kaikki moninpelit eivät käytä edellä kuvattua mallia. Joissain tapauksissa erillisen palvelinsovelluksen tilalla voidaan käyttää asiakassovellukseen integroitua palvelinta, mikä mahdollistaa pelaamisen ja palvelimen hallinnoimisen saman sovelluksen kautta. Tämä saattaa antaa palvelimena toimivalle pelaajalle etulyöntiaseman nopeatempoisissa peleissä, sillä palvelin voi prosessoida paikallisen pelaajan syötteet suoraan sen sijaan, että ne kiertäisivät verkon kautta.

Palvelin-asiakasmallin sijaan voidaan käyttää peer-to-peer-mallia, jossa kaikki pelaajat tekevät pelin tilaan muutoksia ja ilmoittavat niistä kaikille muille. Tällaiset järjestelmät ovat harvinaisia, koska niiden toteuttaminen on erittäin vaikeaa. Peer-to-peer-verkossa kaikki pelaajat joutuvat viestimään kaikkien kanssa ja verkko kuormittuu enemmän. Lisäksi usean erillisen simulaation ajaminen ja

synkronoiminen toisiinsa verkon yli on muun muassa verkkoviiveiden takia vaikeaa.

2.3 Verkkoprotokollat moninpeleissä

Lähes kaikki verkkopelit käyttävät viestintään nykyään IP-verkkoja. Tärkein syy tähän on se, että IP on internetin käyttämä perusprotokolla ja sitä tukevat käytännössä kaikki pelaamiseen soveltuvat laitteet. Jotkin älypuhelimille julkaistut pelit tukevat myös Bluetooth-tiedonsiirtoa. Tähän käytetään yleensä Bluetoothin sarjaporttiprofiilia (SPP, Serial Port Profile).

Kuljetuskerroksen protokollina käytetään yleisimmin UDP:tä. Monissa muissa sovelluksissa yleisemmin käytetty TCP soveltuu huonosti reaaliaikaisiin verkkopeleihin, sillä luotettavan tiedonsiirron toteuttaminen laskee siirtonopeutta ja lisää verkkoviivettä. TCP-pakettien puskurointi ennen lähetystä saattaa pienillä tiedonsiirtomäärillä lisätä viivettä entisestään, mikä voi aiheuttaa sen, että pakettien sisältämä tieto on pahasti vanhentunutta saapuessaan perille. Mikäli luotettavaa tiedonsiirtoa tarvitaan, voidaan UDP:n päälle toteuttaa sovelluksessa kerros, joka varmistaa pakettien pääsyn perille. (13.)

Ylemmillä kerroksilla käytetään pelien vaatimaan nopeaan viestintään soveltuvia protokollia. Nämä protokollat ovat tavallisesti valmistajakohtaisia ja niiden spesifikaatiot ovat harvoin julkisia. Tiettyihin toimintoihin saatetaan kuitenkin käyttää julkisesti määriteltyjä protokollia, kuten XMPP:tä pikaviestintätoimintoihin ja RTP:tä puheäänen kuljetukseen.

2.4 Moninpelien verkkoviestintä

Tässä luvussa tarkastellaan moninpelien verkkoviestintää palvelin-asiakasmallin mukaan toteutettujen pelien näkökulmasta. Harvinaisempia verkkopelijärjestelmien malleja ei käsitellä, koska toteutukset niistä ovat verrattain harvinaisia ja ne sopivat huonosti useimpien tavallisten pelityyppien käyttöön.

Moninpeleissä tarvitaan tiedonsiirtoa ensisijaisesti pitämään pelaajien ja palvelimen versiot pelisimulaation tilasta samanlaisina. Palvelin-asiakasmallissa tämä

tarkoittaa sitä, että palvelin ajaa pelisimulaatiota ja lähettää tietoja sen tilasta asiakkaille. Asiakasohjelman tehtäväksi jää lähinnä paikallisen pelaajan syötteiden prosessointi ja palvelimen lähettämien tietojen esittäminen pelaajalle siten, että pelikokemus on mahdollisimman sujuva. (12.)

Simulaation ajaminen tapahtuu siten, että palvelimelle asiakkailta saapuvat tiedot kerätään puskuriin ja koko puskurin sisältö prosessoidaan kerralla tietyin väliajoin. Palvelimelle saapuvissa tiedoissa on asiakasohjelman luoma aikaleima, joka mahdollistaa syötteiden ja tapahtumien prosessoimisen oikeassa järjestyksessä. Prosessoinnin jälkeen simulaation uusi tila lähetetään kaikille pelaajille. Simulaation päivitysväliä säätämällä voidaan vaikuttaa simulaation tarkkuuteen. Toisaalta on otettava huomioon verkon viive, sillä mikäli datapaketin matka asiakkaalta palvelimelle kestää kauemmin kuin yhden päivitysvälin verran, saattaa syntyä ristiriita.

Voidaan kuvitella esimerkiksi tilanne, jossa pelaajan hahmo on merkitty kuolleeksi yhdellä simulaation päivityskerralla. Tämän jälkeen palvelimelle saapuu matkalla viivästynyt paketti ajalta ennen hahmon kuolemaa, jonka tietojen mukaan pelaaja ehtikin tehdä väistöliikkeen tai jotain muuta sellaista, mikä olisi estänyt pelihahmon kuolemisen. Palvelin joutuu tässä tilanteessa joko jättämään komennon huomiotta tai palaamaan simulaatiossa taaksepäin ja simuloimaan jostain aiemmasta tilasta eteenpäin.

Siirrettävän tiedon määrää voidaan rajoittaa tarkistamalla, millä tiedoilla on merkitystä kullekin pelaajista ja millä taas ei. Tällä on merkitystä erityisesti massiivisille moninpeleille ja virtuaalimaailmoille, joiden palvelimet joutuvat palvelemaan jopa tuhansia pelaajia samanaikaisesti. On järkevää rajata asiakkaalle lähetettävistä viesteistä pois esimerkiksi tiedot liian kaukana tai muutoin asiakkaalta näkymättömissä olevista pelaajista. Tästä merkityksellisen informaation määrittämistä suorittavasta prosessista käytetään nimitystä ”interest management”. (14.)

Toinen verkkoliikenteen määrää vähentävä tekniikka on deltapakkaus. Deltapakkauksessa jonkin objektin koko tilan sijasta lähetetään nykyisen ja edellisen

tilan erotus, joka on usein esimerkiksi peliobjektien sijaintia esittäville koordinaateille hyvin lähellä nollaa. Menetelmän haittapuolena on, että asiakas tarvitsee jokaisen deltapakatun päivityksen tiedot voidakseen päivittää simulaa-tion tilaa luotettavasti. Tällöin on joko käytettävä luotettavaa tiedonsiirtomene-telmää tai luotava deltapakatut päivitykset aina viimeisimmän asiakkaan todistetusti vastaanottaman tilan pohjalta, jolloin hävinneen paketin tiedot sisällytetään automaattisesti seuraavaan lähetettävään päivitykseen (15).

2.5 Verkkoviiveen vaikutukset ja niiden piilottaminen

Tiedonsiirrossa esiintyy aina jonkin verran viivettä. Niin optisessa kuin sähköisessä tiedonsiirrossakin valon nopeus asettaa ehdottoman ylärajan tiedonsiirtonopeudelle, minkä lisäksi jokainen lähettäjän ja vastaanottajan välissä toimiva verkkolaite lisää omasta toiminnastaan johtuvan viiveen (12). Tämä on verkkopelien kannalta tärkeää, sillä pelaajien näytölle piirtyvä näkymä on riippuvainen pelipalvelimen lähettämistä tiedoista.

2.5.1 Simulointi ja ennustaminen asiakassovelluksessa

Aiemmin mainitsin, että verkkopeleissä pelisimulaatio tapahtuu pelipalvelimella ja asiakasohjelman tehtäväksi jää esittää palvelimen lähettämät tiedot. Verkkoviiveen takia aivan näin yksinkertaista järjestelmää ei kuitenkaan voida käyttää kuin hidastempoisissa ja vuoropohjaisissa peleissä, eikä aina niissäkään. Tärkein syy tähän on pelin vasteaika. Mikäli paikallinen sovellus vain lähettäisi syötetyt komennot palvelimelle ja jäisi odottelemaan vastausta tekemättä mitään muuta, pelaaja joutuisi odottamaan koko tämän ajan, ennen kuin hänen komentonsa vaikutukset näkyisivät.

Myös asiakasohjelman on siis tehtävä jonkinlaista simulaatiota, jotta pelaajalle voidaan luoda ainakin uskottava kokemus siitä, että hänen toimintonsa vaikuttavat peliin reaaliajassa. Täysin asiakasohjelman vastuulle simulaatiota ei voida jättää, sillä tämä mahdollistaisi pelissä huijaamisen asiakasohjelman toimintaa muuttamalla. Kompromissiratkaisuna käytetään yleensä jonkinlaista toimintojen vaikutusten ennustamista. Tämä toimii siten, että kun pelaaja antaa pelille ko-

mentoja, asiakasohjelma simuloi niiden vaikutukset pelimaailmaan viimeisimmän palvelimelta saamansa simulaation tilan perusteella. Pelaajalle näytetään tämä ennustettu versio simulaatiosta, jota tarvittaessa korjataan, kun palvelimelta myöhemmin saadaan seuraava simulaation todellinen tila. (16, linkki Client Side Prediction.)

Asiakasohjelman suorittama ennustaminen rajoittuu kuitenkin paikallisen pelaajan toimintoihin ja joihinkin niiden vaikutuksista, sillä muiden pelaajien antamia komentoja ei voida tietää, ennen kuin niistä saadaan palvelimelta tieto. Tässä vaiheessa palvelin on jo laskenut muiden pelaajien komentojen vaiku-tukset pelimaailmaan eikä niitä tarvitse simuloida. Näiden tietojen esittäminen sellaisenaan ei kuitenkaan ole hyvä ratkaisu, sillä pelin kuva päivittyy simu-laation tilaa nopeammin. Tällöin kuvan päivittäminen pelkästään simulaation senhetkistä tilaa käyttäen johtaisi muiden pelaajien ja palvelimen simuloimien objektien nykivään liikkumiseen.

2.5.2 Ekstrapolointi ja interpolointi

Simulaation esittämisen apuna voidaan käyttää ekstrapolointia tai interpolointia. Nämä toimivat hieman eri periaatteilla ja kummassakin on omat ongelmansa, joita ratkotaan eri tavoin. Yhteistä on kuitenkin se, että kummassakin tapauksessa kutakin piirrettävää kuvaa varten lasketaan yhtä tai useampaa tunnettua simulaation tilaa käyttäen arvio simuloitujen objektien sijainnista ja muista tiedoista.

Liikkumisen ekstrapolointiin tarvitaan tiedot objektin sijainnista, suunnasta ja liikenopeudesta. Näiden tietojen perusteella lasketaan objektin oletettu sijainti viimeisimmästä tunnetusta tilasta eteenpäin. Tässä menetelmässä on se ongelma, että liikkeen oletetaan jatkuvan viimeisimpään tunnettuun suuntaan. Todellisuudessa simuloitavien objektien liikesuunta saattaa muuttua hyvinkin nopeasti. Ekstrapolointi saattaa johtaa siihen, että objektit piirretään välillä sellaiseen kohtaan, jossa ne eivät palvelimen mukaan koskaan käyneetkään. Tämä voi aiheuttaa ruudulle piirrettävien objektien ”hyppimistä” paikasta toiseen, kun

simulaation tilaa korjataan palvelimelta tulevien tietojen perusteella. (16, linkki Display of Targets.)

Interpoloinnissa piirrettävä kuva lasketaan kahden tai useamman tunnetun simulaation tilan perusteella siten, että lopputulos esittää arviota objektien tiloista jossain pisteessä näiden tunnettujen tilojen välillä. Tarkan ajankohdan laskemiseen käytetään kaavassa 1 kuvatun kaltaista menetelmää.

$s_0 + dt - id$ *KAAVA 1*

s_0 = viimeisimmän tunnetun tilan ajankohta (s)

dt = s_0:n jälkeen kulunut aika (s)

id = interpolaatioviive (s)

Interpolaatioviive on kiinteästi määritetty viive, joka on yleensä simulaation päivitysvälin suuruinen. Kun interpolaatioviive vähennetään laskuhetken ajankohdasta, saadaan piste kahden tunnetun tilan välissä. Pisteen etäisyys näistä kahdesta tilasta määrää sen, kuinka paljon interpoloidaan.

Menetelmällä saavutetaan ekstrapolaatioon nähden se etu, ettei objekteja piirretä sellaisiin paikkoihin, joissa ne eivät ole olleet. Toisaalta kuvan piirtä-minen interpolaatioviiveen verran menneisyydessä tapahtuneiden asioiden perusteella aiheuttaa ongelmia esimerkiksi tilanteissa, joissa pelaaja yrittää osua liikkuvaan maaliin. Mikäli palvelin ei kompensoi interpolaatioviivettä toimintojen vaikutusta simuloidessaan, on käyttäjän arvioitava itse viiveen suuruus ja tähdättävä liikkuvan maalin eteen osuakseen. (16, linkki Display of Targets.)

Myös interpolaatiosta voi kuitenkin seurata ikävä visuaalinen ongelma. Koska piirrettävät kuvat lasketaan palvelimelta saatujen tilojen perusteella, lopputulos riippuu siitä, mistä kohdasta simulaatiota ja kuinka tiheästi mittapisteet on otettu. Tällöin etenkin ne pisteet, joissa objektin liikkeen suunta muuttuu, saattavat jäädä esittämättä piirrettäessä, mikäli palvelin ei ole sattunut näytteistämään simulaatiota tarkalleen näillä ajanhetkillä. Ongelma tunnetaan nimellä ”bouncing

ball problem” eli pomppivan pallon ongelma (16, linkki Display of Targets). Hyvin samankaltainen ongelma tunnetaan signaalin-käsittelyssä laskostumisena.

2.5.3 Interpolaatioviiveen kompensointi

Nopeatempoisissa peleissä asiakasohjelma käyttää simulaation mittapisteiden välisten tapahtumien esittämiseen useimmiten interpolaatiota, koska ekstrapolaatio aiheuttaa merkittävämpiä vääristymiä ja visuaalisia ongelmia. Interpolaatiossa käytetyn interpolaatioviiveen takia pelinäkymä piirretään kuitenkin aina jonkin verran vanhentuneiden tietojen perusteella, mikä aiheuttaa ongelmia tarkkuutta vaativien toimintojen kuten tähtäyksen kanssa. Näiden ongelmien korjaamiseksi palvelimen on otettava interpolaatioviive jollain tavalla huomioon.

Yksi tapa interpolaatioviiveen poistamiseksi on nk. lag compensation -menetelmä. Menetelmässä kaikkien pelaajien syöttämien komentojen mukana lähetetään aikaleima. Palvelin arvioi kunkin pelaajan verkkoviiveen ja määrittää tämän tiedon perusteella viimeisimmän simulaation päivityksen, jonka pelaaja on todennäköisesti vastaanottanut ennen komennon antamista. Tämän jälkeen lasketaan simulaation tila komennon antamishetkellä ottaen huomioon verkko- ja interpolaatioviiveet. Komennon vaikutukset simulaatioon lasketaan tästä ajankohdasta eteenpäin ja uudet tilat lähetetään normaalisti kaikille pelaajille. (16, linkki Lag Compensation.)

Interpolaatioviiveen poistaminen saattaa aiheuttaa aiemmin kuvatun kaltaisia tilanteita, joissa palvelin joutuu palaamaan simulaatiossa taaksepäin ja muuttamaan jo laskettuja tuloksia. Tulosten muuttaminen on joskus välttä-mätöntä, jotta peliä pystyvät pelaamaan myös ne pelaajat, joilla on suuri verkkoviive palvelimelle. Ilman viiveenpoistoa huonomman yhteyden kautta pelaavat olisivat epätasa-arvoisessa asemassa niiden kanssa, joiden verkko-viive palvelimen kanssa on pieni.

3 UNITY

Unity on pelinkehitysalusta, jolla voidaan kehittää pelejä samanaikaisesti useille eri laitealustoille. Sen kehitys aloitettiin vuonna 2001 ja ensimmäinen julkaisu tehtiin 2005 (17). Unity on saatavilla ilmaisena perusversiona, johon voidaan erillisiä lisenssejä ostamalla liittää erilaisia lisätoimintoja. Ilmaisen version käyttö on yksityishenkilöille ja pienille yrityksille rajoittamatonta. Ehkä juuri ilmaisuutensa ansiosta Unity on tällä hetkellä käytetyimpiä pelimoottoreita (18). Unity oli myös ensimmäisiä iPhone-yhteensopivia pelinkehitystyökaluja, mikä selittää sen suosiota erityisesti mobiilipelien kehityksessä.

Unity koostuu pelimoottorista ja siihen integroidusta editorista. Editorissa luodaan koodi- ja datatiedostoja pelimoottorin komponenttirakenteisiin yhdistelemällä peli, jota pelimoottori suorittaa. Peliä voidaan ajaa suoraan editorissa, mikä mahdollistaa monipuolisen testaamisen ja vaikkapa pelin parametrien muokkauksen ajon aikana. Pelistä voidaan myös kääntää itsenäisesti toimiva binääri- tai asennuspaketti.Unity lukee yleisimmin käytettyjä kuvien, äänten ja 3D-mallien tiedostomuotoja sekä joitain harvinaisempia tiedostomuotoja kuten tracker-musiikkitiedostoja (19). Pakettiin kuuluu myös MonoDevelop, jota käytetään koodin kirjoittamiseen.

Ohjelman kehityksen keskeisenä tavoitteena on ollut luoda työkalu, joka helpottaa yksin tai pienissä tiimeissä toimivien kehittäjien työtä. Editorissa työskentelystä on pyritty tekemään sujuvaa ja monia toimintoja on automa-tisoitu. Kehittäjien työtä helpottaa myös Asset Store, joka toimii eräänlaisena Unityssä käytettävien työkalujen ja tiedostojen kauppapaikkana.

3.1 Keskeiset käsitteet

3.1.1 Unity-projekti ja datatiedostot

Unityllä luodut projektit tallennetaan tietynlaisen hakemistopuun alle. Projektihakemiston juuresta löytyvät koodieditorin omat projektitiedostot, joita Unity käyttää apuna koodia käännettäessä. Sen alla on Assets-hakemisto, jonne tal-

lennetaan kaikki projektiin tuodut datatiedostot sekä pelin ja editorin toiminnallisuutta laajentavat skriptit. Datatiedostoja organisoidaan Assets-hakemiston rakennetta muokkaamalla. Muita projektin hakemistoja ja tiedostoja ei käyttäjän tavallisesti tarvitse muokata itse, vaan kaikki muutokset projektiin tehdään editorin avulla.

Dataa tuodaan projektiin vetämällä tiedostoja hiirellä editoriin tai käyttämällä editorin import-toimintoa. Assets-hakemistoon siirretyt tiedostot editori käy automaattisesti läpi ja lisää projektiin. Kooditiedostot tarkastetaan tuotaessa ja editori ilmoittaa niissä mahdollisesti olevista virheistä. Virheet on korjattava, ennen kuin pelin suorittaminen on mahdollista.

3.1.2 GameObject, Component ja Transform

Kaikkien Unity-pelissä olevien objektien tai entiteettien pohjana on tyyppi GameObject. Sellaisenaan GameObject määrittelee vain objektin sijainnin pelimaailmassa sekä sen kulman ja koon. Sijainti, kulma ja koko eli mittakaava esitetään Transform-tyypin avulla. Transform määrittelee kunkin näistä vektorityyppinä Vector3, joka puolestaan esittää koordinaatteja kolmiulot-teisessa avaruudessa. (20.)

Objektiin saadaan muita ominaisuuksia lisäämällä siihen komponentteja (Component), jotka voivat olla kooditiedostoja eli skriptejä, 3D-malleja, valoja, kameroita, äänilähteitä jne. GameObjecteja voidaan myös asettaa toisten GameObjectien lapsiobjekteiksi. Tämä tekee hierarkisten objektien luomisen mahdolliseksi. Hierarkiassa alemman tason objektit liikkuvat, pyörivät ja skaalautuvat ylemmän tason objektin mukana (20). Tämän ansiosta voidaan GameObjecteja käyttäen luoda esimerkiksi useasta eri osasta koostuva pelihahmo, jonka kaikki osat liikkuvat pelimaailmassa hierarkian ylimmän tason mukana.

Niin GameObjecteja kuin yksittäisiä komponenttejakin voidaan kytkeä pois päältä. Tällöin ne eivät enää näy pelissä eivätkä vaikuta muihin objekteihin. Pois päältä kytkettyjen skriptien koodia ei ajeta. Hierarkian ylimmän objektin kytke-

minen pois päältä vaikuttaa myös sen lapsiobjekteihin muuttamatta kuitenkaan suoraan näiden tilaa. (21.)

3.1.3 Skriptit ja MonoBehaviour

GameObjektien toiminnallisuus luodaan etupäässä liittämällä niihin skriptejä eli Script-tyyppisiä komponentteja. Skriptit ovat kooditiedostoja, jotka sisältävät sen toiminnallisuuden määrittävän luokan. Skriptien luokat eroavat tavallisista luokista siten, että ne perivät Unityn koodikirjaston määrittelemän MonoBehaviour-tyypin. Tiedoston käyttäminen skriptinä edellyttää sen nimeämistä sen sisältämän luokan nimen mukaan, eikä samassa tiedostossa voi olla yhtä useampaa MonoBehaviour-tyypistä johdettua luokkaa. (22.)

MonoBehaviour on tyyppi, jonka periyttäminen luokkaan mahdollistaa skriptin liittämisen GameObjectiin komponenttina. Se määrittelee muutamia funktioita tai metodeja, joilla on pelimoottorille erityinen merkitys (22). Pelimoottori kutsuu näitä funktioita pelin aikana aina tilanteen niin vaatiessa. Esittelen lyhyesti näistä tärkeimmät.

Update on metodi, jota pelimoottori kutsuu ennen jokaisen framen eli ruudulle piirrettävät kuvan luomista (23). Tämän metodin avulla voidaan toteuttaa skriptin toiminnallisuus, kuten käyttäjän syötteiden lukeminen ja objektien liikkuminen pelimaailmassa. Koska ruudun piirtonopeus saattaa vaihdella, Update-metodien kutsujen välissä kuluva aika ei ole vakio. Useat Update-metodissa tehtävät operaatiot onkin järkevää sitoa edellisen kutsun jälkeen kuluneeseen aikaan.

FixedUpdate on puolestaan tasaisin aikavälein kutsuttava metodi. Sillä toteutetaan toimintoja, joiden on tapahduttava aina säännöllisin väliajoin, kuten fysiikan mallinnus. Metodia kutsutaan ennen jokaista fysiikkamoottorin suorittamaa päivitystä (23). Kutsujen välinen kiinteä aika on säädettävissä editorista.

Start ja Awake ovat skriptien alustukseen käytettyjä funktioita. Awake-metodia kutsutaan, kun skripti on ladattu ja alustettu. Start-metodia kutsutaan heti en-

simmäisen framen aikana skriptin lataamisen tai päälle kytkemisen jälkeen ennen ensimmäistä Update-kutsua. (23.)

3.1.4 Scene

Projektin sisältämistä objekteista muodostetaan scenejä, joiden voidaan ajatella vastaavan karkeasti pelin tasoja (24). Todellisuudessa scene on hieman monimutkaisempi käsite. Aina, kun Unity-peli käynnistyy, ladataan määrätty scene, jonka sisältä ohjelman varsinainen toiminnallisuus löytyy. Scene määrittelee kaikki ladattavat objektit sekä niiden alkutilat. Scenejä voi olla projektissa useita. Scenestä toiseen vaihtaminen tapahtuu pelimoottorin LoadLevel-metodeita kutsumalla (25).

Pelin eri tasot voidaan toteuttaa omiin sceneihinsä, mutta scenejä voidaan käyttää myös muihin tarkoituksiin. Tavallista on toteuttaa peliin omana scenenään valikko, joka avautuu pelin käynnistyessä ja josta on mahdollista muuttaa pelin asetuksia ja valita eri tasoja tai pelimuotoja. Scenen ei tarvitse sisältää interaktiivisia osia, joten niitä voidaan käyttää myös esimerkiksi toteuttamaan joukko kiinteitä toimintoja, jotka halutaan aina suorittaa yhdessä.

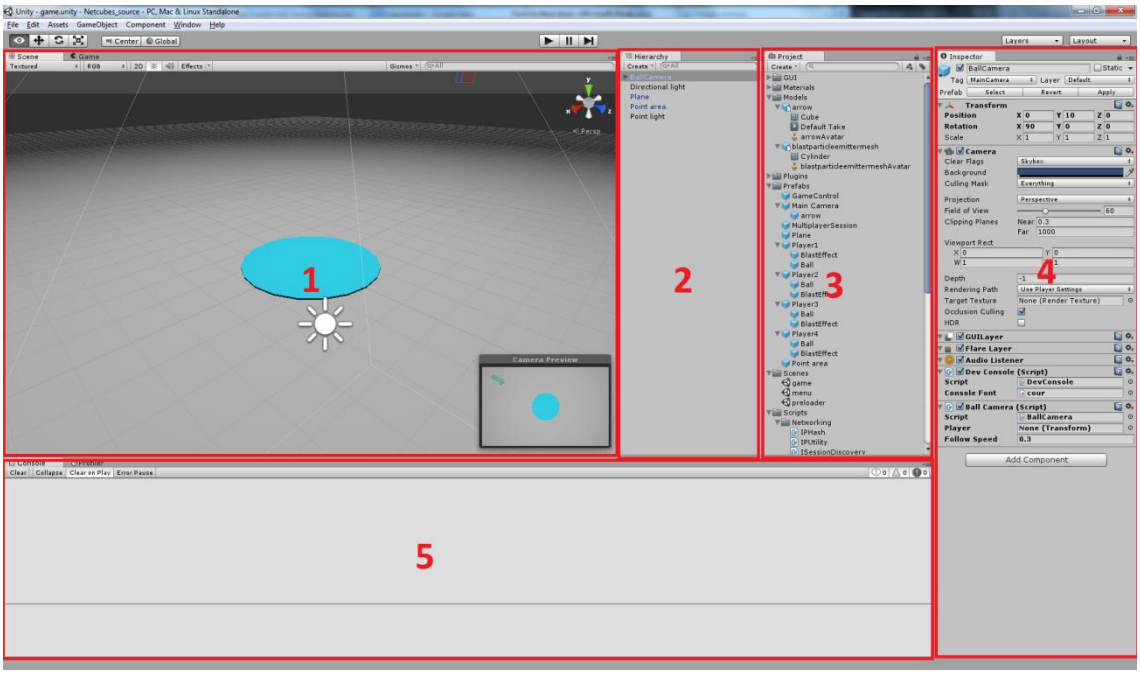
3.1.5 Prefab

Unity-peleissä käytettävät objektit ovat olemassa vain scenen sisällä. Mikäli niitä halutaan siirtää scenestä toiseen, on niistä luotava prefab. Prefab on eräänlainen muotti tai malli, joka kertoo objektin rakenteen ja sen eri osiin kuuluvat komponentit ja niiden parametrit. Prefab luodaan yksinkertaisesti vetämällä objekti editorissa scenen objektilistasta projektitason objektilistaan. Objekteja sen pohjalta luodaan tekemällä sama toimenpide päinvastaiseen suuntaan. (26.)

Prefabiin tehdyt muutokset peilataan automaattisesti kaikkiin sen pohjalta luotuihin objekteihin. Objektin ja prefabin välisen suhteen voi myös halutessaan purkaa, jonka jälkeen muutosten peilausta ei enää tehdä. Mikäli objekteja halutaan luoda tai monistaa koodista käsin on niistä ensin tehtävä prefab.

3.2 Editor

Editor on Unityn pääkäyttöliittymä (kuva 1). Sen avulla datatiedostoista ja skripteistä kootaan pelin scenet, jotka editorin Build-toiminnolla pakataan itsenäisesti ajettavaksi ohjelmaksi. Editorissa peliä voidaan myös testata. (27.)



KUVA 1. Unity-editorin päänäkymä. Numeroidut osat: 1. 3D-näkymä editoriin ladatusta scenestä. 2. Sceneen kuuluvat objektit. 3. Projektin datatiedostot. 4. Inspector, näyttää valitun objektin ja sen komponenttien tiedot ja parametrit. 5. Debug-konsoli.

Editorin tärkein osa on Scene-näkymä, jossa näytetään 3D-esitys kulloinkin ladatusta scenestä. Näkymässä voidaan tarkastella objekteja ja sijoitella niitä paikalleen. Myös esimerkiksi partikkeliefektejä ja erilaisia kuvan piirtämiseen käytettäviä näytönohjaimen suorittamia shader-ohjelmia voidaan tarkastella näkymässä.

Hierarchy-näkymä listaa kulloinkin ladattuna olevat, sceneen kuuluvat objektit. Project-näkymä puolestaan näyttää kaikki projektiin kuuluvat datatiedostot sekä

Unityn luomat rakenteet. Näihin kuuluvat myös projektin scenet, jotka tallennetaan .scene-päätteisinä tiedostoina projektin Assets-hakemistoon. (27.)

Kulloinkin valittuna olevan objektin tiedot näytetään Inspector-näkymässä. Ylimpänä näkymässä on aina objektin nimi, joitain objektin toimintaan liittyviä parametrejä sekä Transform-komponentti. Näiden alta löytyvät kaikki muut komponentit. Inspectorin kautta komponentteja voi poistaa, lisätä ja muokata. Niin objekteja kuin yksittäisiä komponenttejakin voi myös kytkeä päälle ja pois kunkin nimen vierestä löytyvällä valintalaatikolla.

Game-näkymää (kuvassa välilehdessä Scene-näkymän alla) käytetään, kun peli halutaan ajaa editorissa. Se näyttää pelin sellaisena kuin se tulevalle käyttäjälle näkyy. Peliä ajettaessa sceneä voi muokata normaaliin tapaan Scene-näkymän ja Inspectorin kautta, joten peliobjektien muokkaus ja liikuttelu onnistuu pelin suorituksenkin aikana. Näin voidaan tarkastella sellaistenkin objektien toimintaa, jotka eivät itse pelissä näy.

3.3 Skriptaus

Unity-pelien toiminnallisuus toteutetaan liittämällä pelin objekteihin skriptejä, jotka muokkaavat objektien ja pelin tilaa. Skriptejä voidaan kirjoittaa C#-, UnityScript- ja Boo-ohjelmointikielillä. Kaikki nämä käännetään CLI-standardin mukaiseksi tavukoodiksi, jota pelien alustana toimiva Mono-kirjasto suorittaa. (28.)

Skripteillä voidaan myös toteuttaa editoriin uusia toimintoja. Esimerkiksi monimutkaisille ja paljon käytetyille komponenttiskripteille on usein hyödyllistä tehdä erityinen käyttöliittymä, joka helpottaa skriptin parametrien tarkastelua ja muokkaamista. Editorin laajennusskripteillä on pääsy sekä pelimoottorin että editorin toimintoihin.

3.4 Moninpeliominaisuudet

Unityssä on myös RakNet-verkkokirjastolla toteutettuja toimintoja IP-verkkojen yli moninpelattavien pelien toteuttamista varten. Kovin monimutkaisia moninpelejä niillä ei voi toteuttaa, koska säätömahdollisuudet ovat melko rajalliset. Sää-

dettävissä on lähinnä koko sovelluksen tasolla tilasynkronoinnin datapakettien kiinteä lähetysväli, joka ilmaistaan lähetyskertojen määränä sekunnissa. Kaikki verkkoliikenne tapahtuu UDP-protokollaa käyttäen (29).

Erillisen pelipalvelimen käyttöä Unityn omat verkkotoiminnot eivät tue. Yksi pelaajista toimii palvelimena, johon kaikki muut yhdistävät. Palvelin käynnis-tetään yhdellä funktiokutsulla ja yhtä yksinkertaista on siihen yhdistäminen (30). Markkinoilla on kuitenkin useita Unity-yhteensopivia palvelinratkaisuja, joista osa tarjoaa myös editorin kautta säädettävän asiakasrajapinnan.

Unity osaa myös viestiä osoitteenmuunnosta tekevän eli NAT-reitittimen läpi. Tämän ominaisuuden toimiminen riippuu kuitenkin reitittimen asetuksista. Mikäli verkkoviestintä ei näinkään onnistu, voidaan viimeisenä vaihtoehtona reitittää kaikki liikenne kulkemaan välityspalvelimen kautta. (31.)

3.4.1 NetworkView ja tilasynkronointi

Moninpelattavuus itse pelissä toteutetaan lisäämällä objekteihin NetworkView-komponentteja. NetworkView voidaan asettaa tarkkailemaan yhtä objektin muista komponenteista ja lähettämään tiedot muutoksista sen tilassa muille pelaajille. Jokaisella NetworkView-komponentilla on tunnistenumero, joka lähetetään viestien mukana. Tunnistenumero kertoo vastaanottajalle, mille NetworkViewille saapuva viesti on tarkoitettu. Tarkkailtujen komponenttien tietojen lähettämistä kutsutaan tilasynkronoinniksi (engl. state synchronization). (32.)

Synkronointi voidaan tehdä kahdella tavalla (29). Yksi vaihtoehto on lähettää vain tapahtuneiden muutosten suuruudet. Tätä kutsutaan myös delta-pakkaukseksi. Delta-pakattua tietoa siirrettäessä käytetään luotettavaa siirtomenetelmää, koska hukkuneiden delta-pakettien seurauksena lähettäjän ja vastaanottajan pelien tilat erkanisivat toisistaan.

Toinen vaihtoehto on lähettää koko komponentin tila aina kun tietoa siirretään (29). Tämä käyttää hieman enemmän tiedonsiirtokapasiteettia kuin delta-pakattujen tietojen lähettäminen. Matkalla hukkuneita paketteja ei kuitenkaan

tarvitse lähettää uudelleen, sillä paketit ovat toisistaan riippumattomia ja vastaanottaja pystyy päivittämään tietonsa oikeaan tilaan minkä tahansa vastaanotetun paketin perusteella. Päivityspakettien katoaminen matkalla saattaa aiheuttaa vastaanottajan näkymässä visuaalisia ongelmia, kuten objektien äkillistä liikahtelua paikasta toiseen. Tämä ei kuitenkaan vaikuta itse pelin toimintaan.

3.4.2 Remote procedurecall

Tilasynkronoinnin lisäksi NetworkViewin kautta voidaan tehdä etäproseduurikutsuja (engl. remote procedure call). Tällöin käsky kutsua tiettyä skriptin metodia sekä metodin parametrit lähetetään verkon yli. Kutsuja voidaan lähettää joko kaikille pelaajille samanaikaisesti tai rajatummin vain osalle pelaajista. Etäproseduurikutsuihin liittyvä data lähetetään heti, kun kutsu on tehty, eikä synkronointipakettien päivitysväli vaikuta niihin. (33.)

Metodit, joita halutaan kutsua verkon yli, on merkittävä koodissa RPC-määreellä. Näin täytyy tehdä siksi, että Unity osaa lähettää verkon yli vain muutamia datatyyppejä. Ohjelmaa käännettäessä RPC-määreellä merkittyjen metodien parametrilistat käydään läpi, ja mikäli parametreissä on käytetty vääriä tyyppejä ilmoitetaan virheestä käyttäjälle. Varsinaiset etäproseduuri-kutsut tehdään NetworkView-komponentin RPC-metodia kutsumalla. Metodi ottaa vastaan kutsuttavan etäproseduurin nimen, kutsun vastaanottajat määrittävän RPCMode-parametrin sekä lopuksi kutsuttavan proseduurin parametrit. (33.)

3.4.3 OnSerializeNetworkView

Sellaisenaan NetworkView osaa synkronoida Transform-, Animation- ja Rigidbody-tyyppisten komponenttien tietoja (29). Mikäli halutaan synkronoida itse luotuja komponentteja on tietojen lähettämiseen ja vastaanottamiseen tarvittava koodi kirjoitettava MonoBehaviourin OnSerializeNetworkView-metodiin (34). Järjestelmä kutsuu metodia automaattisesti jokaisella verkon päivitysvälillä ainakin kerran.

Metodin parametrit ovat BitStream ja NetworkMessageInfo. BitStream on objekti, jonka metodien kautta tietoja voidaan lähettää ja vastaanottaa. Lisäksi se sisältää kentät, jotka kertovat, ollaanko tietoja lähettämässä vai vastaanottamassa. NetworkMessageInfo on tietorakenne, joka sisältää kentät lähettäjän ja tiedot lähettäneen NetworkViewin tunnistamiseksi sekä datapaketin aikaleiman.

4 VERKKOMONINPELIN KEHITTÄMINEN UNITY-ALUSTALLA

4.1 Lähtökohdat

Opinnäytetyön ensimmäisen vaiheen tavoitteena oli kehittää Unityä käyttäen verkon yli pelattava moninpeli. Pelin tulisi toimia ilman erillistä palvelinta, mikä käytännössä tarkoittaa sitä, että palvelimen tehtäviä hoitaa yksi pelissä mukana olevista päätelaitteista. Itse peli haluttiin pitää yksinkertaisena, koska tavoitteena oli tutustua nimenomaan verkkopelien ohjelmointiin eikä peliohjelmointiin sinänsä.

Syntyi ajatus eräänlaisesta kukkulan kuningas -variantista, jossa pelaajat ohjaavat tasaisella pelikentällä pyöriviä palloja tarkoituksenaan pysyä peli-kentän keskellä sijaitsevalla pistealueella. Lähimpänä pistealueen keskustaa oleva pelaaja saa pisteitä ja peli päättyy, kun yksi pelaajista saavuttaa vaaditun pistemäärän. Pallojen ohjaus tapahtuu tökkimällä palloja haluttuun suuntaan. Idea vaikutti sopivan yksinkertaiselta tähän projektiin, mutta toisaalta sellaiselta, että toiminnallisuutta voidaan haluttaessa laajentaa ja alustan eri ominaisuuksia päästään idean puitteissa kokeilemaan kattavasti.

Päätettiin myös, että valinnaisena lisäominaisuutena peliin voidaan toteuttaa automaattinen pelisessioiden etsiminen verkosta. Tähänkään ei tarvittaisi palvelinta, vaan sessioiden hakuun voidaan käyttää esimerkiksi UDP-protokollan broadcast-viestejä. Näin hakupyynnöt välitettäisiin automaattisesti kaikille verkkoon kuuluville päätelaitteille.

Lähdin kehittämään peliä ensisijaisesti Android-laitteille. Mobiilipelien ohjelmointi oli minulle uutta ja mahdollisuus kokeilla erilaisia tekniikoita kiinnosti. Unity tekee pelien julkaisemista eri laitealustoille todella helppoa, joten Android-version rinnalla pelistä tehtiin testaamisen helpottamiseksi myös Windows-versio.

4.2 Suunnittelu

Pelin suunnitteluun varattiin aikaa viikko. Tänä aikana syntyivät ensimmäinen versio pelin verkko-osien ohjelmistoarkkitehtuurista sekä yksinpelattava prototyyppi itse pelistä. Prototyyppiin toteutettu toiminnallisuus siirtyi valmiiseen moninpelattavaan versioon lähes sellaisenaan, mutta ensimmäinen suunnitelma verkko-osien rakenteesta osoittautui melko kömpelöksi käyttää. Se suunniteltiinkin toteutusvaiheessa kokonaan uudestaan. Ohjelman käyttöliittymä suunniteltiin mahdollisimman yksinkertaiseksi, jotta sen toteuttamiseen ei tarvitsisi käyttää paljon aikaa.

Verkkopelitoiminnallisuuden toteuttamiseksi harkittiin kahta eri vaihtoehtoa. Unityssä on yksinkertainen ja helppokäyttöinen verkkokerros, joka on toiminnoiltaan melko rajoittunut mutta riittävä monenlaisten verkkopelien toteuttamiseen. Mahdolliseksi korvaajaksi tälle harkittiin AllJoyn-verkkokirjastoa (35), jolla voidaan yhdistää erilaisia päätelaitteita toisiinsa WiFin tai Bluetoothin yli. AllJoyn-verkkokirjastosta on saatavilla myös Unity-yhteensopiva versio.

Peli päädyttiin kuitenkin toteuttamaan Unityn omilla verkkotoiminnoilla, sillä AllJoynin käyttäminen olisi kasvattanut sovelluksen oman verkkokoodin määrää merkittävästi. Lisäksi opinnäytetyön toisessa vaiheessa tultaisiin vertailemaan erilaisia Unity-yhteensopivia pelipalvelimia, ja Unityn omien verkkotoimintojen tunteminen saattaisi olla tällöinkin eduksi. Olin jo aiemmin kokeillut UDP broadcast -viestien käyttämistä samassa lähiverkossa toimivien päätelaitteiden automaattiseen yhdistämiseen, joten päätin käyttää menetelmää myös tässä projektissa, mikäli aikaa jäisi valinnaisten osien tekemiseen.

Työn tilaajan pelit on suunnattu nuorille lapsille ja tämä haluttiin ottaa huomioon myös opinnäytetyössä. Ohjaajan aloitteesta kehitettiin IP-osoitteistukseen perustuva yksinkertainen osoitejärjestelmä, jossa IP-osoitteesta pudotetaan verkko-osa pois ja jäljelle jäävästä osoiteosasta muodostetaan merkkijono, joka ilmaistaan ikoneina. Kun yksi pelaaja syöttää omaan laitteeseensa toisen pelaajan ruudulla näkyvät ikonit, sovellus laskee näitä vastaavan IP-osoitteen ja koettaa yhdistää siihen.

4.3 Toteutusvaiheet

Pelin toteutuksen pohjana käytettiin suunnitteluvaiheessa tehtyä prototyyppiä, jolla pelin perustoiminnallisuutta eli pallon liikuttelua kentällä ja pisteiden laskentaa oli kokeiltu. Liikkumiseen käytetään Unityn omaa fysiikkamoottoria, joka oli havaittu prototyyppivaiheessa tarkoitukseen mainiosti sopivaksi. Valmiilla fysiikkamoottorilla pallot saatiin helposti paitsi liikkumaan myös tönimään toisiaan.

Pallojen pyörimisliikkeen havaittiin tuottavan fysiikkamoottorille ongelmia. Vaikka pallojen fysiikkakomponentti oli asetettu pitämään pallot pystyakselin suhteen paikoillaan, pyöriminen aiheutti liikkeeseen ikävää pomppimista. Tämän seurauksena pallot saattoivat jopa hävitä hetkellisesti pelikentän alle. Ongelma ratkaistiin siten, että fysiikkakomponentit säädettiin pitämään myös pallojen asento vakiona. Pelissä näkyvä pyörimisliike toteutettiin pallon ruudun päivityskertojen välillä kulkeman matkan perusteella laskettavana visuaalisena efektinä.

Fysiikkamoottorin avulla peliin toteutettiin myös eräänlainen räjäytystoiminto. Pelinäkymään lisättyä Blast-painiketta painamalla pelaajan pallo ”räjähtää” sinkauttaen muita lähellä olevia pelaajien palloja kauemmaksi. Samalla näyte-tään yksinkertaisena partikkeliefektinä toteutettu räjähdysefekti. Toiminnon käyttämistä rajattiin lisäämällä käyttökertojen väliin viive, jottei peli muuttuisi liian kaoottiseksi.

Seuraavaksi toteutettiin valikoista koostuva käyttöliittymä, jonka avulla pelin järjestäminen ja peliin liittyminen tapahtuvat. Koska pelin kohdealustana toimivista Android-laitteista löytyy laaja kirjo erilaisia näyttöjä toteutettiin käyttöliittymään toiminto, joka skaalaa fonttien koot ja käyttöliittymän osat erilaisille näytöille sopiviksi. Samaa koodia käytetään myös pelinäkymän käyttöliittymäkomponenttien skaalaamiseen.

Valikkokäyttöliittymän avulla ohjelmoitiin verkkokirjasto, joka toteuttaa pelisession pelaajien välille. Kirjaston tehtävänä on huolehtia yhteydenmuodostuksesta, yhteyden tilasta sekä session tietojen, kuten pelaajien nimien ja yhteyksien tilojen, välittämisestä kaikille sessioon kuuluville. Yksi pelaajista aloit-

taa session ja muut liittyvät siihen, jonka jälkeen kommunikointiin käytetään Unityn verkkotoimintoja.

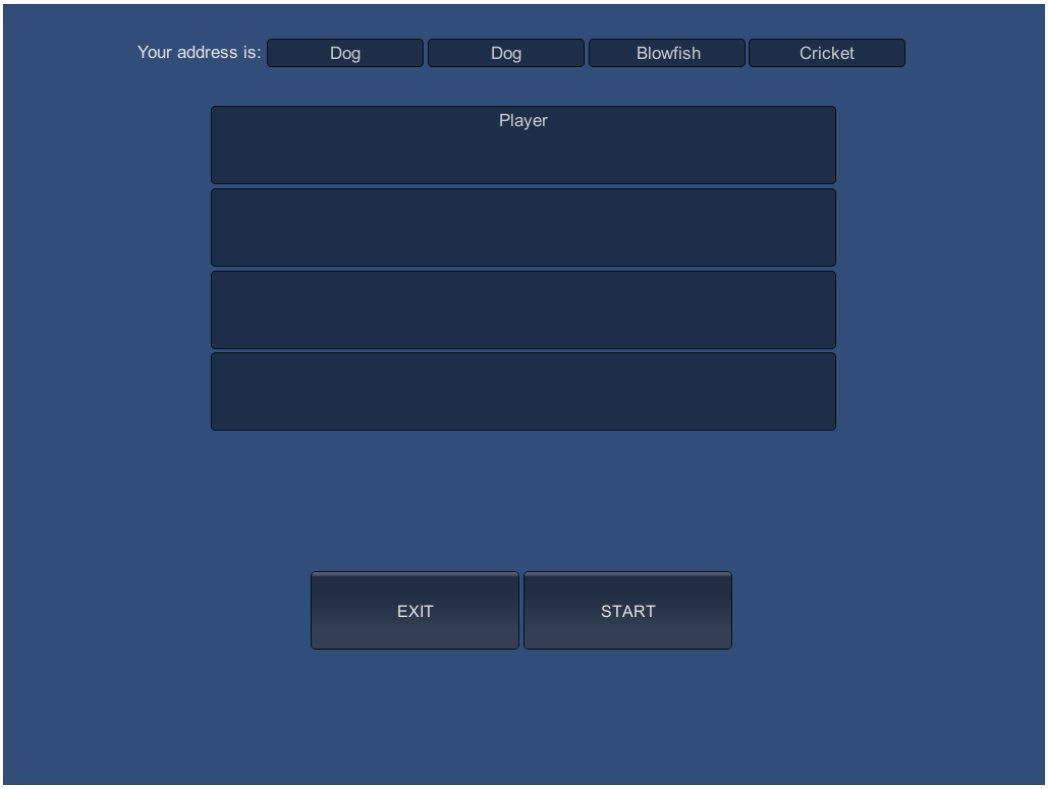
Kun yhteydenmuodostus ja muut kirjaston toiminnot oli saatu toimimaan halutulla tavalla yhdistettiin käyttöliittymä peliin ja itse peliä alettiin muokata moninpelattavaksi. Pallojen liikkeet synkronoitiin pelaajien välillä NetworkView-komponenteilla Unreliable-asetusta käyttäen. Epäluotettavaan tiedonsiirtoon päädyttiin, kun kumpaakin vaihtoehtoa oli testattu ja havaittu, että mobiililaitteella WiFi-verkon yli pelattaessa luotettavaa tiedonsiirtoa käyttävä synkronointi tuotti pallojen liikkeisiin enemmän ”nykimistä”. On syytä kuitenkin huomata, että toimintaa testattiin yrityksen verkossa, jossa oli samaan aikaan paljon muutakin liikennettä. Paremmissa verkko-olosuhteissa tuloksetkin olisivat saattaneet olla paljon parempia. Pistetilanteen synkronoimiseen taas käytettiin NetworkViewin luotettavaa delta-pakattua tiedonsiirtoa. Tämä käyttää hieman vähemmän tiedonsiirtokaistaa kuin koko objektin tilan lähettäminen jokaisella päivitysvälillä.

Synkronointiviestien lisäksi tarvittiin signalointia pelin alkamisesta ja päättymisestä tiedottamaan sekä Blast-toimintoa varten. Nämä toteutettiin etäproseduurikutsuilla. Blast-toiminnon osalta palvelin-asiakasmallista poikettiin sikäli, että asiakassovellukset pitävät itse kirjaa toiminnon käyttöä rajoittavasta ajastimesta, eikä palvelin tiedä ajastimesta mitään. Lisäksi asiakassovellus lähettää itse kaikille pelaajille käskyn näyttää räjähdysefekti. Tässäkin tapauksessa palvelin vastaa kuitenkin fysiikan simuloimisesta.

Moninpelattavuuden toteuttamisen jälkeen tehtiin enää joitain säätöjä ja kosmeettisia muutoksia. Pelinäkymään lisättiin nuoli, joka osoittaa aina kentän keskustaa kohti. Pallon sijainnin hahmottamista helpottamaan pistealueen yläpuolelle sijoitettiin pistemäinen valonlähde, jonka voimakkuus heikkenee pistealueesta etäännyttäessä. Pallojen liikuttelua säädettiin siten, että pelin tuntuma olisi mahdollisimman samankaltainen mobiililaitteen kosketusnäytöllä ja PC:n hiirellä pelattaessa.

4.4 Sovelluksen toiminta

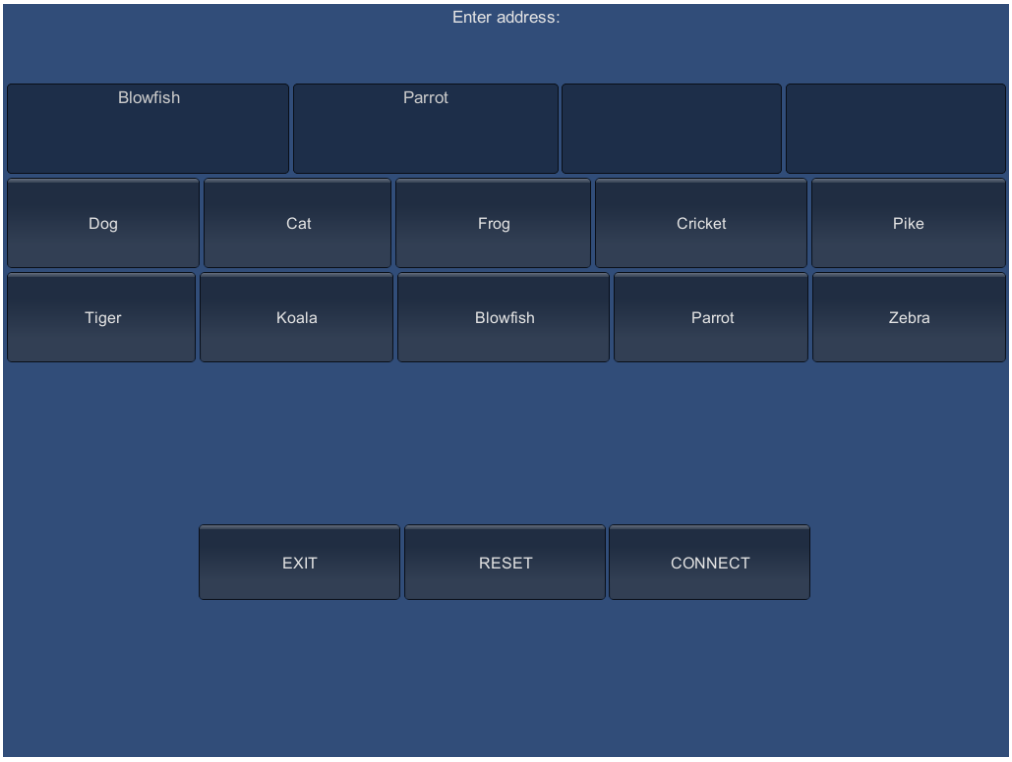
Pelin käynnistyttyä näytetään ruudulla päävalikko, jossa on kolme painiketta. Näistä ensimmäinen, Host Game, käynnistää pelipalvelimen ja vie pelaajan ns. Lobby-näkymään (kuva 2), jossa on lista pelisessioon liittyneistä pelaajista ja laitteen osoitetta esittävät ikonit. Lobby-näkymässä on kaksi painiketta, joista toinen palauttaa pelaajan päävalikkoon ja toinen aloittaa varsinaisen pelin. Host Game -painiketta painettaessa käynnistyy myös prosessi, joka lähettää verkkoon säännöllisin väliajoin tiedon palvelimen olemassaolosta.



KUVA 2. Lobby-näkymä. Ylälaidassa pelaajan verkko-osoitetta esittävät ikonit, sen alla lista pelissä olevista pelaajista.

Toinen painike puolestaan avaa Join Game -näkymän (kuva 3), johon osoitteen voi syöttää. Näkymän Connect-painikkeen painaminen aloittaa yhdistämisprosessin, jonka päätyttyä näytetään sama Lobby-näkymä kuin session aloittaneelle pelaajalle eli session isännälle. Pelin aloittava Start-painike on kuitenkin

vain isännän ruudulla. Mikäli palvelimeen ei saada yhteyttä näytetään Lobby-näkymän sijaan virheviesti.

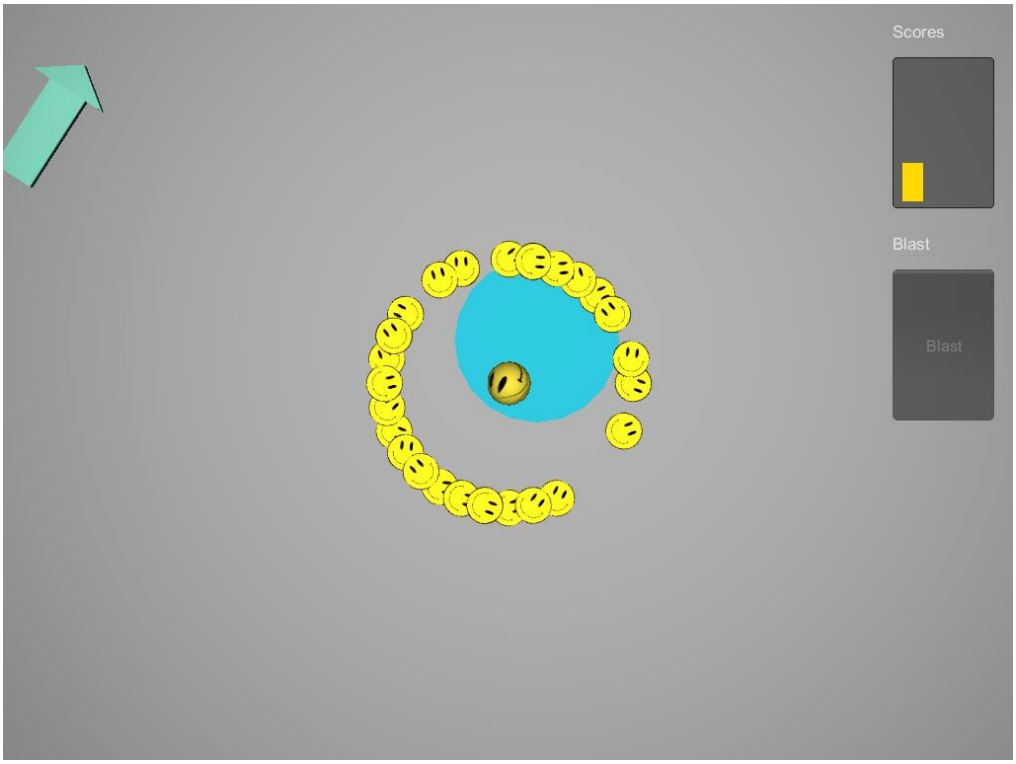


KUVA 3. Join Game -näkymä. Ylimpänä laatikot, joilla syötetty osoite ilmaistaan. Näiden alapuolella painikkeet osoitteen syöttämistä varten.

Päävalikon kolmas painike käynnistää automaattisen palvelimen hakuprosessin. Hakuprosessi lähettää säännöllisin väliajoin verkkoon broadcast -viestillä tiedon siitä, että sovellus on hakutilassa. Samalla se kuuntelee verkosta palvelinten lähettämiä viestejä. Mikäli Lobby-tilassa oleva palvelin vastaanottaa viestin hakutilassa olevalta sovellukselta, se yrittää lähettää suoraan tälle tiedon omasta sessiostaan. Asiakassovellus listaa löytämänsä sessiot hakutilan näkymään, ja kun käyttäjä valitsee jonkin niistä, sovellus yhdistää palvelimeen normaalisti ja siirtyy Lobby-tilaan.

Pelinäkymään (kuva 4) siirtymisen jälkeen ruudulla näkyvät lyhyen aikaa ohjeet pelin pelaamiseksi sekä laskuri, jonka laskettua nollaan itse peli alkaa. Pelinäkymän oikeassa yläkulmassa on laatikko, jossa värillisillä palkeilla esitetään

pelaajien pistemäärät. Sen alapuolella on Blast-painike, jonka painaminen lennättää lähellä olevat pelaajat kauemmaksi. Tämä toiminnallisuus on kunkin pelaajan käytettävissä vasta viiden sekunnin kuluttua pelin alkamisesta tai edellisestä käyttökerrasta.



KUVA4. Pelinäkymä. Pelaaja on juuri käyttänyt Blast-toimintoa. Keltaiset kiekot ovat osa räjähdysefektiä.

Pelikenttä on taso, jonka keskellä on ympyrän muotoinen pistealue. Kamera kuvaa peliä suoraan ylhäältä. Pelaajat liikuttavat palloa ruudun päällä haluttuun liikesuuntaan pyyhkäisemällä tai PC-versiossa tekemällä vastaavan liikkeen hiirellä. Kun joku pelaajista saavuttaa vaaditun pistemäärän, peli päättyy ja kaikille näytetään lopulliset tulokset. Tästä näkymästä pääsee Exit-painikkeella takaisin päävalikkoon.

4.5 Verkko-osien rakenne ja toiminta

Pelin verkkotoiminnot toteutettiin Unityn verkkokomponentteja sekä C#-kielen System.Net-kirjastoa käyttäen. Unityn tarjoamat toiminnot ovat melko yksinkertaisia, ja heti projektin alussa päädyinkin rakentamaan niiden ympärille kerroksen, joka parantaa verkon tilan hallintaa ja helpottaa uusien toimintojen toteuttamista. Alkuperäinen suunnitelma muistutti kuitenkin lopulta toteutunutta mallia vain etäisesti.

Toteutetun verkkokerroksen tärkein käsite on sessio. Sessiolla tarkoitetaan tilaa, jossa yhden tai useamman pelaajan välillä on verkkoyhteys ja jossa he näin kykenevät viestimään keskenään. Tapaan, jolla tämä viestintä tehdään ei sessiokerros periaatteessa ota kantaa. Käytännössä kuitenkin käytössä olevat viestintätavat ovat riippuvaisia siitä, kuinka yhteys on muodostettu. Tämän takia sessiokerros jaettiin kahteen osaan.

MultiplayerSession on abstrakti luokka, joka määrittelee sessiokerroksen rajapinnan ja toiminnot session tilan hallitsemiksi. Se toteuttaa kaikki ne kerroksen toiminnot, jotka ovat riippumattomia käytetystä tiedonsiirto-menetelmästä. Kaikkien pelaajien saatavilla on tämän luokan kautta lista sessioon kuuluvista pelaajista ja näiden tiedoista. Sessiokerroksen kautta kaikille pelaajille välittyvät myös ilmoitukset session tapahtumista, kuten liittyvistä ja poistuvista pelaajista. Ilmoitusten välittämiseen käytetään C#-kielen Event-rakenteita.

StandardMultiplayerSession puolestaan perii MultiplayerSession-luokan ja toteuttaa sessiokerroksen tiedonsiirtomenetelmästä riippuvaiset toiminnot käyttäen Unityn tarjoamia verkkotoimintoja. Varsinaiseen yhteydenmuodos-tukseen liittyvä toiminnallisuus peilaa melko tarkasti Unityn toimintoja: session aloittamiseen ja palvelimen käynnistämiseen käytettävä Initiate-metodi kutsuu Unityn Network.InitializeServer-metodia ja sessioon liittymiseen käytettävä Connect puolestaan kutsuu Network.Connect-metodia. Yhteyden tilaan ja mahdollisiin virhetilanteisiin liittyvää käsittelyä on lisätty. Sessioon liittyvien tietojen välittämiseen käytetään etäproseduurikutsuja.

MultiplayerSession-luokan lisäksi StandardMultiplayerSession toteuttaa ISessionDiscovery-rajapinnan. Tämän rajapinnan tehtävänä on automaattinen pelisessioiden haku verkosta. Tämä tapahtuu aiemmin mainituilla UDP-protokollan broadcast-viesteillä. Session aloittaneen pelaajan asiakasohjelma lähettää verkkoon viestejä, joissa kerrotaan session nimi. Sessiota etsivän pelaajan ohjelma kuuntelee näitä viestejä verkosta ja muodostaa saapuneiden viestien perusteella listan saatavilla olevista sessioista. Broadcast-viestissä on aina lähettäjän IP-osoite, joten sitä ei tarvitse lähettää erikseen.

StandardMultiplayerSession-luokan toteutuksessa rajapinnan StartAdvertisingSession-metodi avaa portin kuuntelua varten ja alkaa lähettää verkkoon säännöllisin väliajoin viestejä, joilla session olemassaolo ilmaistaan. StartSessionDiscovery-metodia vastaavasti käytetään, kun halutaan kuunnella näitä viestejä verkosta. Kummassakin tapauksessa viestien kuunteluun käytetään samaa porttia.

Peliohjelman tapauksessa sessiota aletaan mainostaa verkossa, kun pelaaja käynnistää palvelimen Host Game -painikkeesta. Samasta valikosta löytyvä Find Game -painike asettaa sessionhakutoiminnon kuuntelutilaan. Painikkeesta avautuvaan näkymään listataan löytyvät sessiot omina painikkeinaan. Näistä pelaaja voi valita session, johon sovellus yrittää painalluksen jälkeen liittyä normaalisti.

5 MONINPELIPALVELIMET UNITY-ALUSTALLE

Tässä luvussa luodaan katsaus markkinoilla oleviin virtuaalimaailmojen toteuttamiseen soveltuviin ja Unityn kanssa yhteensopiviin palvelinratkaisuihin. Palvelimia tutkittiin ja testattiin opinnäytetyön toisessa vaiheessa marras-joulukuussa 2013. Tuotteita pyrittiin vertailemaan kattavasti vertailukelpoisten ominaisuuksien osalta sekä testaamaan ainakin sen verran, että niiden tärkeimmistä ominaisuuksista sekä toimintaperiaatteista saatiin muodostettua hyvä käsitys.

Tarkastelussa pyrittiin kiinnittämään huomiota erityisesti tuotteiden soveltuvuuteen pienelle pelialan yritykselle ja pienen kehittäjätiimin kannalta tärkeisiin ominaisuuksiin. Monen yhtäaikaisen käyttäjän virtuaalimaailman toteuttaminen on jo sinällään huomattavan suuri urakka, eikä pienen yrityksen rajallisia resursseja ole syytä tuhlata tarkoitukseen huonosti sopivan tuotteen kanssa kamppailemiseen.

5.1 Tutkittavien tuotteiden valinta

Palvelinkatsaus aloitettiin tekemällä kartoitus sellaisista markkinoilla olevista tuotteista, joilla online-virtuaalimaailmoja voidaan toteuttaa. Suunnitelmana oli käydä läpi mahdollisimman monen palvelinohjelmiston tiedot ja valita näistä noin viisi sopivinta lähempään tarkasteluun. Määrää rajattiin, jotta kaikki mukaan valitut tuotteet ehdittäisiin tutkimaan ja testaamaan riittävän kattavasti. Ensisijaiset kriteerit jatkoon valinnalle olivat tuotteen kypsyys, kehitysaktiivisuus ja soveltuvuus mainittuun käyttötarkoitukseen. Tietolähteinä valintaa suoritettaessa käytettiin tuotteiden dokumentaatiota, kotisivuja ja sekä kehittäjien että muiden toimijoiden ylläpitämiä keskustelupalstoja.

Tuotteen kypsyys tarkoittaa tässä lähinnä sitä, että luvatut ominaisuudet on toteutettu ja tuote toimii luotettavasti. Dokumentaation laatu ja laajuus olivat kypsyyttä arvioitaessa tärkeässä osassa: puutteellinen tai virheellinen dokumentaatio viittaa siihen, että itse tuotekin saattaa olla keskeneräinen tai ettei kaikkia ominaisuuksia ole testattu kunnolla. Dokumentaation laadusta voitaneen myös

päätellä jotain siitä, kuinka halukkaita ja/tai kyvykkäitä kehittäjät ovat tukemaan tuotettaan.

Kehitysaktiivisuutta voidaan pitää sikäli tärkeänä valintakriteerinä, että tuote, jota ei kehitetä eteenpäin saattaa lakata toimimasta vaikkapa sen käyttämien ohjelmistokirjastojen päivittyessä ja rajapintojen muuttuessa. Internetin kautta jaeltavien ohjelmistojen kanssa on myös hyvä muistaa, että ohjelmisto saattaa olla saatavissa jotain kautta vielä kauan senkin jälkeen, kun kehittäjä on sen hylännyt. Tällaisessa tilanteessa tuotteen saatavuuden äkillinen lakkaaminen on mahdollista. Pienelle yritykselle riski siitä, että sen käyttämän tuotteen tuki loppuu varoittamatta on liian suuri otettavaksi. Kehitysaktiivisuuden määrittämiseksi käytiin läpi tuotteiden julkaisu- ja päivityshistorioita sekä tuotteeseen liittyviä uutisia ja tiedotteita.

Erilaisilta keskustelupalstoilta haettiin tutkinnan alla olevia tuotteita käsitteleviä keskusteluita. Nimimerkeillä käytävän verkkokeskustelun luotettavuutta ei kuitenkaan voida pitää kovin korkeana. Näiden keskusteluiden tutkimisen tarkoituksena olikin ensisijaisesti muodostaa jonkinlainen käsitys siitä, kuinka laajalti mikäkin tuote pelinkehittäjien keskuudessa tunnetaan ja millaisia sovelluksia niillä on toteutettu. Lisäksi esimerkiksi kehittäjille tällaisten palveluiden kautta osoitetut tukipyynnöt ja kysymykset sekä vastaukset näihin voivat antaa jotain viitteitä siitä, kuinka valmis tuote on ja kuinka aktiivisesti kehittäjät sitä tukevat.

Palvelinohjelmiston soveltuvuus nimenomaan virtuaalimaailmojen toteuttamiseen on valintakriteereistä vaikein määrittää. Periaatteessa lähes millä tahansa palvelinohjelmistolla voidaan toteuttaa jonkinlainen virtuaali-maailma. Valinnoissa otettiin huomioon sellaiset ohjelmistoihin toteutetut ominaisuudet, jotka tavalla tai toisella tukevat virtuaalimaailmapelejä. Tällaisia ominaisuuksia ovat esimerkiksi palvelimeen yhdistetty tietokanta ja sisäänrakennetut kuormantasaustoiminnot tai mahdollinen tuki ulkoisille vastaaville toiminnoille suurten käyttäjämäärien palvelemisen mahdollistamiseksi. Myös kehittäjien ilmoittama soveltuvuus eri käyttötarkoi-tuksiin sekä eri alustoja käyttäen jo toteutetut tuotteet otettiin tässä huomioon.

Kaikkia näitä kriteerejä arvioitaessa otettiin huomioon paitsi palvelinohjelmisto myös sen tukemat asiakasrajapinnat. Varsinaisen palvelintuotteen lisäksi peliohjelmiston kehittäjä tarvitsee rajapinnan, jonka kautta palvelimen toimintoja käytetään asiakassovelluksesta. Tähän tarkoitukseen tarjotaan yleensä jonkinlainen SDK (software development kit) eli ohjelmistonkehityspaketti, joka sisältää joko binääri- tai lähdekoodimuodossa palvelimen kanssa viestimiseen tarvittavat toiminnot. Unity-yhteensopivan SDK:n olemassaolo oli yksi edellytyksistä tuotteen valinnalle jatkokierrokselle. Joko osana ohjelmiston-kehityspakettia tai sen lisäksi tarjolla on myös lähes poikkeuksetta jonkinlainen toimiva esimerkkisovellus.

Noin viikon kestäneen kartoituksen jälkeen päädyttiin yhteensä viiteen eri palvelinratkaisuun, joita vertailtaisiin tietyiltä osin sekä kokeiltaisiin mahdolli-suuksien mukaan. Tuotteet pyrittiin valitsemaan siten, että ne edustavat saatavilla olevaa tarjontaa mahdollisimman kattavasti. Joukkoon valittiin niin perinteisiä palvelinohjelmistoja kuin yksi pilvipalvelukin. Kaikista valituista tuotteista on saatavilla jonkinlainen ilmainen kokeiluversio, mikä on perus-edellytys sille, että tuotteita voidaan tällaisessa työssä kunnolla vertailla.

5.2 Menetelmät ja vertailukriteerit

Tuotevertailua suoritettiin tuotteiden teknisten ominaisuuksien, ylläpitotoimintojen ja muiden ominaisuuksien kuten lisenssiehtojen osalta. Alkuvaiheessa koottiin mahdollisimman monesta potentiaalisesta vertailukriteeristä lista, josta myöhemmin karsittiin vähäpätöisempiä ja vaikeimmin vertailtavia kohtia pois. Lopullisen listan perusteella laadittiin vertailua varten Excel-taulukko (liite 1). Tähän taulukkoon kerättiin vertailun aikana tuotteista kerätyt tiedot ja se toimi projektin tärkeimpänä apuvälineenä.

Tietoja keräämällä suoritetun vertailun lisäksi suurinta osaa valituista tuotteista kokeiltiin käytännössä. Tässä olennaisimmaksi välineeksi muodostuivat tuotteiden dokumentaatiot sekä valmiit esimerkkiprojektit, jotka esittelevät tuotteen toimintaa. Esimerkkiprojektien avulla palvelinten toimintaa voitiin kokeilla käyttämättä aikaa asiakassovellusten toteuttamiseen. Koska useimpia vertailuun

valituista projekteista markkinoidaan Unity-yhteensopivina saatiin esimerkkiprojektien avulla myös nopeasti jonkinlainen kuva siitä, kuinka hyvin yhteensopivuus todella on toteutettu.

5.3 Vertaillut tuotteet

5.3.1 ElectroServer 5

ElectroServer on perinteinen pelipalvelinohjelmisto, joka toimii Windows-, Linux-, Unix- ja OSX-alustoilla (36). Ohjelmiston uusin versio on Electroserver 5, joka tarjoaa kehitysrajapinnat Unityn lisäksi käytetyimmille web- ja mobiilialustoille (38). Saatavilla on lisäksi vanhempaan nelosversioon perustuva Electrotank Universe Platform eli EUP, joka on virtuaalimaailmojen ja massiivisten moninpelien luomiseen tarkoitettu ohjelmistopaketti (37). EUP olisi ollut tämän projektin tavoitteita ajatellen uudempaa Electroserveriä mielenkiintoisempi vaihtoehto testattavaksi, mutta koska siitä ei ole saatavilla kokeiluversiota ei tämä ollut mahdollista.

Palvelinohjelmisto toimii Java-alustan päällä. Palvelimessa ei ole omia paikallisesti käytettäviä hallinnointityökaluja, vaan kaikki hallinta ja säädöt ohjelmiston ensimmäistä käynnistystä lukuunottamatta tehdään erillisen etähallintatyökalun kautta. Tämä ratkaisu vaikutti kokeilukäytön perusteella hyvältä. Hallintaohjelmistossa on kattavasti ominaisuuksia palvelimen säätämistä ja monitorointia, käyttäjien hallinnointia sekä esimerkiksi chat-viestien suodatusasetusten muokkausta varten.

Palvelimen toiminnallisuutta voidaan laajentaa Java-, JavaScript- ja ActionScript-moduuleilla. Moduulit lähetetään etäpalvelimelle joko hallintatyökalujen kautta tai FTP:tä käyttäen. Palvelinta paikallisena sovelluksena ajettaessa on myös mahdollista kopioida laajennuksen hakemistorakenne palvelimen extensions-hakemistoon. Palvelin lataa laajennukset automaattisesti käynnistyessään ja kirjoittaa mahdollisesti syntyneet virheilmoitukset lokitiedostoon. Kehittäjät suosittelevat JavaScript- ja ActionScript-moduulien käyttämistä vain kehitystarkoituksiin, sillä Java-moduuleja pystytään suorittamaan paljon nopeammin.

Ohjelmiston mukana tulee kymmenen valmista esimerkkisovellusta ja useita kymmeniä pienempiä koodiesimerkkejä. Kaikkien esimerkkien palvelinkomponentit on toteutettu Javalla ja asiakassovellukset näille vaihtelevasti joko Flash-, Java-, Unity- tai JavaScript-sovelluksina. Esimerkeistä kokeiltavaksi valittiin Reversi, joka toteuttaa yksin- ja moninpelattavan version Suomessa paremmin Othellona tunnetusta lautapelistä. Sovelluksen asentaminen palvelimelle oli helppoa ja palvelimen uudelleenkäynnistyksen jälkeen peli oli jo pelattavissa. Pelaamaan päästiin Unityllä toteutetun asiakassovelluksen kautta.

Unity-sovellusta tutkiessa esille nousi yksi ElectroServerin varjopuolista: integraatio editorin kanssa jättää toivomisen varaa. ElectroServer toimii Unityn kanssa yhteen siten, että sovelluksessa luodaan yksi ElectroServer-tyyppinen olio, joka tarjoaa koko palvelinrajapinnan toiminnot. Rajapintaa käytetään kaikista muista pelisovelluksen osista tämän yhden olion metodeja kutsumalla. Tämä mahdollistaa palvelinrajapinnan laajan mukauttamisen lähes kaiken-laisten sovellusten tarpeisiin, muttei toisaalta istu kovin hyvin yhteen Unity-sovellusten komponentteihin perustuvan rakenteen kanssa. Jonkinlaiset valmiit verkkotoiminnallisuuden toteuttavat MonoBehaviour-komponentit ja esimerkiksi näiden kautta tapahtuva koordinaattitietojen automaattinen synkronointi palvelimen ja asiakkaiden välillä helpottaisi palvelinrajapinnan käyttämistä.

ElectroServer vaikuttaa melko valmiilta ja vakaalta palvelinohjelmistolta. Alun perin Flash-sovellusten kanssa yhteensopivaksi palvelimeksi tarkoitettu tuote on saatu kiitettävästi toimimaan yhteen nykyaikaisempien pelinkehitysteknologioiden kanssa. Tuotteen pitkä historia on valitettavasti havaittavissa niin hyvässä kuin pahassakin. Yhteensopivuus Unity-editorinkanssa on toteutettu vain melko alkeellisella tasolla eikä palvelimen toiminnallisuutta ole mahdollista laajentaa C#-kielellä toteutetuilla moduuleilla. Viimeisin ominaisuus on ehkä puutteista pahin, sillä suuri osa Unity-projekteista kirjoitetaan nimenomaan C#:a käyttäen. Eri ohjelmointikielen käyttäminen palvelimen päässä aiheuttaa sen, että kaikki sellainen koodi, jota täytyy suorittaa sekä asiakkaan että palvelimen puolella on kirjoitettava kahteen kertaan.

Tuotteen tulevaisuuden kannalta mielenkiintoisena yksityiskohtana verkossa pelattavia kasinopelejä kehittävän High 5 Gamesin mukaan se on sopinut ostavansa kaikki oikeudet ElectroServerin kehittäjän, ElectroTankin tuotteisiin. Samalla kaikkien ElectroTankin työntekijöiden on määrä siirtyä ostajan palvelukseen (39). Tästä sopimuksesta ei kuitenkaan mainita mitään ElectroTankin omassa tiedotuksessa. ElectroServerin myynti ilmeisesti jatkuu toistaiseksi normaalisti, sillä yritys tarjoaa edelleen aktiivisesti tukea tuotteidensa käyttäjille tukifoorumin välityksellä. Uusin päivitetty versio ElectroServeristä on julkaistu ilmeisesti huhtikuussa 2013 (40), joten on epäselvää kehitetäänkö tuotetta edelleen.

5.3.2 FrostNet

FrostNetiä markkinoidaan uudenlaisena ratkaisuna toteuttaa reaaliaikaisia moninpelejä. Toiminnaltaan se on melko perinteinen Windows- ja Linux-alustoilla toimiva palvelinohjelmisto, joka on tehty erityisesti HTML5-teknologiaa käyttävien Unity-pelien kehittämistä varten (41). FrostNetin kehitys aloitettiin vuonna 2012, joten se on vertailun tuotteista tuorein (42).

Ohjelmistossa on joitain sellaisia ominaisuuksia, joita ei muista tuotteista löydy ja joiden ansiosta se valikoitui vertailuun. Tuote tarjoaa ainakin teoriassa helpon tavan hankkia uusia pelaajia sen päälle kehitetyille peleille. Tämä perustuu siihen, että Unityllä kehitetyistä peleistä voidaan FrostNetin työkaluilla tehdä webselaimessa pelattava HTML5-versio. Näin potentiaaliset uudet pelaajat pääsevät helposti ja nopeasti kokeilemaan peliä. HTML5-version avulla voidaan pelata moninpelejä yhdessä muille alustoille käännettyjen asiakassovellusten kanssa. (41.)

FrostNet tukee palvelin- ja asiakaskoodin kehittämistä samanaikaisesti Unityn avulla. Palvelimen ja asiakkaan kesken jaetuista C#-kooditiedostoista käännetään palvelinohjelmistoa varten oma koodikirjasto, joka annetaan palvelimelle käynnistettäessä komentoriviparametrien välityksellä. Kääntäminen tapahtuu Unityyn integroitua Monoa ja FrostNetin tarjoamia editor-laajennuksia käyttäen, joten erillisiä työkaluja ei tarvita. Palvelimen asetuksia voidaan säätää Unityn

kautta ja editor-laajennukset mahdollistavat myös verkko-ongelmien simuloimisen ongelmatilanteita ratkottaessa. Näiden työkalujen käyttäminen oli vaivatonta ja ne vaikuttivat hyvin toimivilta.

Unityn oman NetworkViewin synkronointia vastaava toiminnallisuus saadaan lisäämällä peliobjektiin Actor-komponentti. FrostNet tarjoaa myös mekanismin etäproseduurikutsujen tekemiseen. Ohjelman kotisivuilta saatava yksinkertainen esimerkkiprojekti demonstroi näitä ominaisuuksia. Sen palvelin luo yksinkertaisen virtuaalimaailman, jossa voidaan liikutella hahmoja.

FrostNet vaikuttaa mielenkiintoiselta tuotteelta, mutta siinä on tiettyjä vakavia puutteita. Merkittävimmät puutteet löytyvät dokumentaatiosta (43), joka on pahasti keskeneräinen. Suurta osaa ohjelmiston ominaisuuksista ei ole lainkaan dokumentoitu ja niiden hyödyntäminen on näin melko vaikeaa. Niiltä osin, joilta dokumentaatiota on saatavilla on sen laatu vaihtelevaa. Osasta ominaisuuksista kerrotaan melko ylimalkaisesti paikoitellen jopa niin, että ainoa tieto mitä niistä on saatavilla on se, että ominaisuuden pitäisi ohjelmistosta löytyä.

Tuotteen kotisivuilla keskitytään esittelemään mahdollisuutta kääntää Unity-sovellukset HTML5-alustalla toimiviksi. Tämän pitäisi onnistua erillistä työkalua käyttäen. Työkalua ei kuitenkaan ole saatavilla, mikä tuntuu hieman harhaanjohtavalta. Toisaalta HTML5-teknologian standardointi on vielä kesken (44), joten tässä vaiheessa Unityn oman Web player-teknologian käyttäminen voi olla kannattavampi vaihtoehto selaimessa toimivien pelien toteuttamiseksi.

Tuotteen kehitysaktiivisuus ei myöskään vaikuta kovin lupaavalta. Dokumentaation mukaan ensimmäinen ja toistaiseksi viimeisin julkinen versio kehitystyökaluista on julkaistu elokuussa 2013. Kaikesta päätellen dokumentaatiotakaan ei ole päivitetty tämän jälkeen. Roadmapissa (45) lupaillaan uusia ominaisuuksia alkuvuodelle 2014, mutta tämänkään viimeisestä päivitysajankohdasta ei ole mitään tietoa.

FrostNet jättää ristiriitaisen vaikutelman. Palvelimessa on hyviä ominaisuuksia, jotka ainakin testatuilta osin toimivat niin kuin on luvattukin. Niin esimerkki-

projekti kuin työkalutkin ovat hyvin tehtyjä ja mahdollistavat tuotteen kokeilemisen. Dokumentaation onneton laatu ja mysteeriksi jäänyt kehitysaktiivisuus eivät kuitenkaan anna mahdollisuutta suositella tuotetta vakavaan käyttöön. Lisenssiehdoista tai hinnoista ei tuotteen kotisivuilla puhuta mitään, mikä osaltaan vahvistaa vaikutelmaa siitä, että FrostNet on vasta kehityksensä alkutaipaleella.

5.3.3 Photon Server

Photon on ExitGamesin kehittämä ja markkinoima tuoteperhe, johon kuuluu useita erilaisia palvelinohjelmistoja. Tuotteista vanhimman, palvelinohjelmisto Photon Serverin (46) lisäksi tarjolla on erilaisia kuukausihinnoiteltuja pilvipalvelupaketteja. Lisäksi erillisenä tuotteena löytyy Photon Chat, joka on chat-palveluiden toteuttamiseen tarkoitettu pilvipalvelu.

Virtuaalimaailmojen toteuttamiseen Photonin pilvipalveluversiot eivät kuitenkaan sovellu, koska ainakaan toistaiseksi pilvipalvelinten toiminnallisuus ei ole ohjelmoitavissa (47). Tätä perustellaan sillä, että mikäli virtuaalipalvelimella ajettavassa asiakkaan koodissa on bugi, se saattaa kaataa koko fyysisen palvelimen ja kaikki muiden asiakkaiden kyseisellä palvelimella ajettavat ohjelmistot siinä sivussa. Photon Realtime-tuotteen esittelyssä mainitaan ”CustomLogic Option”, jonka voisi olettaa tarkoittavan palvelimen ajaman koodin muokkaamista, mutta tästä ei löydy tarkempaa tietoa mistään.

Keskitytään siis tarkastelemaan Photon Serveriä, joka on perinteinen ohjelmistopakettina myytävä tuote. Palvelinohjelmisto toimii Windows-alustalla ja on saatavilla joko kuukausihinnoiteltuna tai kerralla maksettavalla lisenssillä. Kummassakin tapauksessa hinta on maksettava jokaiselta käytettävältä sovellukselta tai palvelimelta ja hinnoittelu on porrastettu siten, että tuki suuremmalle määrälle yhtäaikaisia käyttäjiä nostaa hintaa. Saatavilla on lisäksi kuukausihinnoiteltu Enterprise-lisenssi, joka mahdollistaa rajattoman määrän palvelimia ja käyttäjiä kiinteällä kuukausimaksulla. (48.)

Photonia mainostetaan markkinoiden yhteensopivimpana palvelinratkaisuna. Lista yhteensopivista alustoista käsittää kaikki tärkeimmät pelinkehitykseen käytetyt mobiili-, web- ja työpöytäalustat sekä joitain vähemmän tunnettujakin. Yhteensopivuuden laatu kuitenkin vaihtelee hieman tuoteperheen eri jäsenten välillä. Siinä missä pilvipalvelupohjaiset Photon-tuotteet tarjoavat paljolti Unityn omien verkkotoimintojen tapaan toimivat peliobjekteihin liitettävät komponentit, Photon Server jättää hieman enemmän käyttäjänsä tehtäväksi. Tällä saavutetaan parempi mukautettavuus erilaisiin sovelluksiin, mutta ohjelmiston käyttämiseen projekteissa tarvitaan hieman enemmän työtä. Jonkin verran valmiita komponentteja löytyy palvelinohjelmiston mukana toimitettavista esimerkkiprojekteista, ja osa niistä vaikuttaa käyttökelpoiselta sellaisenaankin.

Palvelimen ohjelmointikielenä toimii C# ja palvelinsovelluksen ajamiseen käytetään Microsoftin .NET-alustaa. Palvelimen kanssa viestitään Photonin kirjaston tarjoaman Peer-luokan avulla. Yhteydenmuodostus ja kaikki viestintä tapahtuu tämän luokan metodeja käyttämällä. Varsinainen tiedonsiirto suoritetaan kutsumalla palvelimen koodissa määritettyjä tapahtumia tai eventtejä Unityn etäproseduurikutsuja muistuttavan järjestelmän kautta. Toisin kuin etäproseduurikutsujen tapauksessa kaikki lähetettävä data pakataan HashTable-tyyppiseen taulukkoon. Tiedon vastaanottaminen palvelimelta tapahtuu samalla menetelmällä.

Palvelinsovellusta käytetään PhotonControl-hallintaohjelman kautta. Valikkopohjainen hallintaohjelma tarjoaa pelipalvelinten hallintamahdollisuuden lisäksi toiminnot lisenssien aktivoimista ja poistamista sekä erilaisten statistiikkatietojen ja lokitiedostojen tarkastelua varten. Varsinaisen pelipalvelimet voidaan käynnistää joko tavallisina sovelluksina aktiivisen käyttäjän oikeuksilla tai Windows-palveluina, jolloin palvelinta ajetaan pääjärjestelmäkäyttäjän oikeuksilla. Palvelin on kummassakin tapauksessa käyttöliittymätön, joten ainoa tapa hallita sitä on hallintaohjelma.

Erikoisuutena Photonissa on sisäänrakennettuna toiminto, jolla pelaajien aiheuttama kuormitus pystytään jakamaan useamman pelipalvelimen kesken. Tätä varten yksi palvelin asetetaan nk. master-palvelimeksi, joka pitää kirjaa kaikista

järjestelmään kuuluvista pelipalvelimista ja ohjaa pelaajilta saapuvat yhteydenmuodostuspyynnöt kulloinkin vähiten kuormitetulle palvelimelle. Pelipalvelimet ilmoittavat itse olemassaolostaan master-palvelimelle, jonka osoite ohjelmoidaan näiden asetuksiin kiinteästi.

Tuote on kattavasti dokumentoitu. Kotisivujen kautta pääsee käsiksi hieman sekavaan verkkodokumentaatioon (49), josta löytyy liuta tutoriaaleja, esimerkkisovellusten kuvaukset sekä paljon sekalaista tietoa tuotteen käytöstä. Lisäksi SDK:n mukana toimitetaan PDF-muotoinen referenssidokumentti, joka sisältää kaikkien koodikirjaston sisältämien luokkien ja toimintojen tiedot ja rajapinnat.

Verkkodokumentaatiosta käy ilmi, että viimeisin päivitys tuotteeseen on julkaistu huhtikuussa 2012 (50). Päivityksen on myös ilmoitettu olevan vasta julkaisukandidaatti Photon Serverin kolmannesta pääversiosta. Aivan täysin valmiilta tuote ei tämän takia vielä vaikuta, eikä sen tulevaisuudennäkymistä ole varmaa tietoa.

Vaikka kehittäjien painopiste vaikuttaakin siirtyneen Photon Serveristä tuoteperheen pilvipalvelutuotteisiin on Photon edelleen melko varteenotettava vaihtoehto muiden Unity-yhteensopivien pelipalvelinten rinnalla. Tuote on erinomaisesti tuettu ja kehittäjät ratkovat aktiivisesti käyttäjien ongelmia. Haittapuolina Unity-integraatio ei ole aivan huipputasoa eikä tuen jatkuminen ole nykyisen julkaisutahdin valossa itsestään selvää.

5.3.4 Player.IO

Ainoa vertailuun valittu pilvipalvelutuote on Player.IO. Tuotetta markkinoidaan erityisesti selaimessa toimivien sekä mobiilipelien toteuttamiseen sopivana palvelinratkaisuna (51). Ilmeisesti tätä tarkoitusta silmälläpitäen tuotteeseen on yhdistetty mahdollisuus käyttää Facebookin ja Kongregate-pelisivuston autentikointitoimintoja käyttäjien tunnistamiseen (52) sekä maksujärjestelmä pelin sisällä tehtäviä mikromaksuja varten (53).

Rajoittuneen ilmaisversion lisäksi tuotteesta on tarjolla Plus-, Pro- ja Enterprise-lisenssitasot. Plus ja Pro ovat kuukausihinnoiteltuja, kun taas Enterprise-lisenssit neuvotellaan erikseen. Kullakin tasolla on rajoitukset mm. tiedonsiirto-määrille, etäpalvelimen levytilalle sekä yhtäaikaisten käyttäjien ja tietokanta-objektien määrälle. PayVault-maksujärjestelmää ei ilmaisen version kanssa voi käyttää lainkaan, korkeammilla lisenssitasoilla sen peritään maksu jokaista maksutapahtumaa kohti. Kalleimmilla lisenssitasoilla on saatavissa myös yhden käyttäjän käyttöön varattu palvelinkone. Ilmaisen ja Plus-lisenssin omistajat joutuvat tyytymään jaettuun palvelinresurssiin.

Palvelinta tarjotaan pelkästään pilvipalveluna, joten asiakkaan palvelimilla ajettavaa tuotantokäyttöön tarkoitettua palvelinsovellusta ei ole saatavilla. Ohjelmistonkehityspaketin mukana tulee kuitenkin myös palvelinsovellus, joka mahdollistaa pelin testaamisen paikallisesti. Kehityspalvelimen ominaisuuksia ei päästy kokeilemaan esimerkkiprojektiin ja dokumentaatioon liittyvistä ongelmista johtuen. Näistä lisää tuonnempana.

Pelien ja palvelinten hallinta tapahtuu tuotteen kotisivujen kautta. Sisäänkirjautumisen jälkeen avautuu hallintasivu, joka näyttää listan käyttäjän rekisteröimistä peleistä sekä tilastoja tiedonsiirtokaistan ja tietokannan käyttöasteeseen liittyen. Hallintasivuston kautta pääsee tarkastelemaan yksittäisten pelin tietoja, kuten käyttäjämääriä, maksutapahtumia ja palvelinresurssien käyttöä. Myös kehityspalvelinten tietoja voi katsella. Käyttöliittymä on hyvin suunniteltu ja toimintojen käyttäminen on helppoa ja nopeaa.

Palveluun on integroitu tietokantapalvelin. Tietokanta on nk. NoSQL-tietokanta, eli se poikkeaa toiminnaltaan ja rakenteeltaan perinteisemmistä relaatiotietokannoista. Kirjoittajalle NoSQL-tietokannat eivät olleet ennestään tuttuja, mutta tämä ei tuottanut ongelmia. Tietokantaobjekteja ja tauluja lisätään ja muokataan web-hallintapaneelin kautta. Systeemi on muiden hallintapaneelin tapaan helppokäyttöinen ja tietokanta on erikoisesta rakenteestaan huolimatta intuitiivinen ja sen käyttö nopeasti opittavissa.

Palvelimella ajettavan pelin koodin kirjoittamiseen voi käyttää mitä tahansa Microsoftin .NET-alustan tukemaa kieltä. Esimerkkiprojektit on toteutettu C#-kielellä. Dll-tiedostoiksi käännetyt koodikirjastot siirretään pilveen web-hallintapaneelia käyttäen.

Kehitystyökalut mahdollistavat Flash-, .NET- ja Unity-pelien kehittämisen. Tuotteen kotisivuilla ja dokumentaatiossa Flash on näistä selvästi enemmän esillä, mikä kertonee jotain tuotteen historiasta. Unity-ja .NET-rajapinnat on kuitenkin hyvin dokumentoitu, ja useimpia ominaisuuksia esittelevää esimerkkikoodia löytyy niin ActionScript- kuin C#-kielilläkin.

Tuotetta oli tarkoitus testata SDK:n mukana tulevaa esimerkkiprojektia ja kehityspalvelinta käyttäen, mutta näitä ei koskaan saatu toimimaan. Palvelin-koodi kääntyi ongelmitta ja kehityspalvelin vaikutti toimivan kuten pitikin.Unity-sovellusta ei kuitenkaan lukuisista yrityksistä huolimatta saatu yhdistämään kehityspalvelimeen eikä pilveen. Ohjelmakirjaston sekavista virheilmoituksista ei ollut mitään apua, ja kehityspalvelimen dokumentaatiokin jättää toivomisen varaa.

Player.IO vaikuttaa lupaavalta tuotteelta. Ominaisuuksia ei ole valtavasti, mutta toteutetut ominaisuudet mahdollistavat ainakin yksinkertaisempien pelien luomisen. Palvelinresurssien automaattinen skaalautuminen käytön mukaan kuulostaa hyvältä ominaisuudelta, mutta testausvaiheessa kohdatuista ongelmista johtuen jäi epäselväksi, kuinka hyvin tämä toimii. Dokumentaatiossa on joitain aukkoja, mutta näitä paikkaa kehittäjien aktiivisuus keskustelupalstalla esitettyihin kysymyksiin vastaamisessa.

Tuotteen kehitysaktiivisuudesta ei ole tietoa, sillä keskustelupalstalla (54) ei ole kerrottu uusista julkaisuista vuoden 2010 jälkeen. Kehityspalvelimen julkaisuhistoriasta löytyy kuitenkin maininta yhteensopivuudesta Unityn vuoden 2012 lopulla julkaistun version 4 kanssa, joten ainakin jotain kehitystä on tehty vielä tuolloin. Player.IO:n kotisivuilla kerrotaan, että Yahoo! on ostanut tuotetta kehittävän yrityksen (55). Kehitystiimiä laajennetaan aktiivisesti, minkä voisi olettaa vaikuttavan positiivisesti sekä tuotteen kehitysvauhtiin että tuen saatavuuteen.

5.3.5 UnityParkSuite ja uLink

UnityParkSuite on MuchDifferentin kehittämä kokoelma Unity-verkkopelien kehittämiseen tarkoitettuja työkaluja. Näistä tärkein, uLink, on verkkokerros, joka on tarkoitettu korvaamaan Unityn verkkotoiminnot. Se poikkeaa vertailun muista tuotteista siinä mielessä, ettei siihen sisälly omaa palvelinohjelmistoa. Sen sijaan sekä palvelin- että asiakassovellus ovat Unity-ohjelmia. (56.)

uLinkin lisäksi UnityParkSuiteen kuuluu muutamia muita itsenäisiä osia. uGameDB on verkkopelien taustajärjestelmäksi suunniteltu hajautettu tietokantapalvelin. uZone ja uPikko ovat useasta palvelimesta koostuvan järjestelmän hallinnointiin ja kuormituksen tasapainotukseen tarkoitettuja työkaluja. Testaustyökalu uTsungilla palvelimia voidaan kuormittaa ja testata automaattisesti. (57.)

Tuotteen käyttäminen kehittämistarkoituksessa on ilmaista. Ilmaisversion kokeiluaika on periaatteessa rajattu 30 päivään, mutta tätä ei ole teknisesti rajoitettu, joten kokeilua on mahdollista jatkaa rajattomasti. Rajoittamattoman käytön mahdollistavan lisenssin voi ostaa 550 eurolla, mikäli vuotuinen liikevaihto jää alle sadan tuhannen euron. Tätä suurempien yritysten täytyy neuvotella lisenssisopimus erikseen (58). Hintaa selittää osaltaan se, että tuotetta kehittää voittoa tavoittelematon yhdistys.

Koska paketti ei sisällä omaa palvelinsovellusta on uLink täysin riippuvainen Unitystä. Toisaalta samasta syystä sen yhteensopivuus Unityn kanssa on vertailun parhaimpia. Rakenteeltaan uLink on siinä määrin samankaltainen Unityn verkkokomponenttien kanssa, että valmiin Unityllä toteutetun verkkopelin verkkokomponentit voidaan lähes suoraan korvata uLinkin vastaavilla. Muutoksia täytyy tällöin tehdä lähinnä kooditiedostoihin. Kaikki nämä toiminnot voidaan tehdä automaattisesti uLinkiin kuuluvaa työkalua käyttäen.

uLinkin riippuvuus Unitystä on kaksiteräinen miekka. Verkkosovellusten kehittämistä varmasti helpottaa ja nopeuttaa se, että palvelin ja asiakassovellus voivat käyttää samoja kooditiedostoja. Samalla mahdollisten virheiden määrä vähenee, kun toimintoja ei tarvitse toteuttaa erikseen kumpaakin päätä varten.

Unityä ei ole kuitenkaan suunniteltu palvelinohjelmistoksi, joten on hieman kyseenalaista kuinka hyvin se tähän tarkoitukseen soveltuu etenkään suurten pelaajamäärien kanssa. MuchDifferentin tammikuussa 2012 järjestämässä tapahtumassa 999 pelaajaa pelasi samaa peliä selaimessa toimivilla asiakasohjelmilla, joten ainakin jonkinlaista näyttöä järjestelmän toimivuudessa vaativammissakin tilanteissa on (59).

Testaukseen olisi ollut mielenkiintoista käyttää opinnäytetyön edellisessä vaiheessa toteutettua pelisovellusta. Työkalu, jolla Unityn verkkokomponenttien korvaamisen uLinkin vastaavilla olisi pitänyt käydä automaattisesti ei kuitenkaan tehnyt yhtään muutosta projektiin tai kooditiedostoihin, eikä aikaa näiden muutosten tekemiseen käsin ollut. Näin ollen testauksessa päädyttiin tekemään valmiin malliprojektin avulla.

Malliprojekti toteuttaa yksinkertaisen lumisotaa simuloivan moninpelin. Pelin tasoista on palvelinta ja asiakasta varten toteutettu omat scenensä. Molempien testaaminen onnistuu editorissa, ja Unityn Build-toiminnolla voidaan oikeat scenet valitsemalla kääntää itsenäisesti toimivat palvelin- ja asiakassovellukset. Unity-sovellukset voidaan käynnistysparametrien avulla ajaa myös komentorivisovelluksina, joten palvelinsovelluksen ajaminen onnistuu myös pelkän komentorivikäyttöliittymän tarjoavalla palvelinkäyttöjärjestelmällä. Testausta ajatellen palvelinsovelluksen ajaminen graafisena on hyvä vaihtoehto.

Rakenteeltaan uLinkin verkkokomponentit muistuttavat todella paljon Unityn omia komponentteja. NetworkView-komponentin korvaa uLinkin saman niminen komponentti, ja verkkokirjaston C#-rajapinta on identtinen Unityn rajapinnan kanssa lukuisia lisättyjä toimintoja lukuunottamatta. uLinkin komponenteissa on kuitenkin enemmän säätömahdollisuuksia. Komponenttien omien säätöjen lisäksi uLink lisää editoriin valikon, jonka kautta voi säätää verkkokerroksen toimintaa koko sovelluksen tasolla.

Tuotteen dokumentaatio on vertailun parhaimmistoa. Kirjaston luokat ja rakenteet esittelevä referenssidokumentaatio (60) on kattava ja tieto on pääosin ajantasaista. Eri toimintojen toiminta ja käyttötarkoitukset on esitelty selkeästi ma-

nuaalissa (61). Näiden lisäksi löytyy joukko tutoriaaleja, joissa vaihe vaiheelta kerrotaan, kuinka erilaisia toimintoja toteutetaan.

UnityPark Suite vaikuttaa kypsältä ja laadukkaalta tuotteelta. Toiminnallisuudesta ei löytynyt merkittäviä vikoja tai puutteita ja dokumentaatiokin on erinomaista. Tuotetta myös kehitetään aktiivisesti, ja kehittäjät ratkovat käyttäjien ongelmia keskustelupalstan välityksellä.

5.4 Yhteenveto

Vertailuun otettiin varsin erilaisia tuotteita, koska haluttiin luoda mahdollisimman kattava katsaus markkinoihin. Mukaan valittiin vain Unity-yhteensopivia palvelimia, sillä tietoa etsittiin juuri Unity-pelin suunnittelun avuksi. Tämä rajoitti vertailuun soveltuvien tuotteiden joukkoa vain vähän, sillä useimmissa markkinoilla olevissa pelipalvelinohjelmistoissa on tätä nykyä jonkinlainen tuki Unitylle.

Tapoja toteuttaa tämä yhteensopivuus on yhtä monta kuin palvelimiakin. Yhtä ääripäätä edustava UnityPark toimii kokonaan Unityn varassa ja on säädettävissä etupäässä tämän editorin kautta. Toiset tuotteet taas tarjoavat pelkän koodirajapinnan binäärimuotoiseen kirjastoon. Tässä suhteessa palvelimen valinta on lähinnä makukysymys, sillä pelin toimintalogiikka ei ole riippuvainen palvelimesta ja sen toteuttamiseen kuluu näin ollen suurin piirtein sama aika palvelimesta riippumatta. Myös rajapintojen dokumentaatioiden laatu vaihtelee, ja tällä on jo enemmän merkitystä tuotetta valittaessa.

Joissain tuotteissa on enemmän ominaisuuksia kuin toisissa, mutta kaikista löytyvät samat pelipalvelimen perustoiminnot. Tavallisia lisätoimintoja ovat esimerkiksi integroitu tietokantapalvelin ja pikaviestintäjärjestelmä. Tällaiset lisätoiminnot voivat helpottaa kehitystyötä, mutta ne eivät välttämättä sovellu kaikkiin käyttökohteisiin ominaisuuksien tai suorituskyvyn puutteen vuoksi.

Suorituskyvyn osalta palvelinohjelmistoja ei vertailtu. Pelilogiikan laskennan suoritusteho on enimmäkseen riippuvainen palvelinkoneen laitteistosta ja käytetystä ohjelmistoalustasta, joka on tyypillisesti joko .NET, Java tai Mono. Tiedon-

siirtokyvyn vertailu erityisesti itse ylläpidettävien palvelinohjelmistojen ja pilvi-palveluiden välillä olisi ollut mielenkiintoista, mutta tähän ei valitettavasti ollut aikaa.

6 LOPPUSANAT

Tällä opinnäytetyöllä oli kaksi päätavoitetta: toteuttaa moninpelattava peli Unity-pelinkehitysalustaa käyttäen sekä tutkia, vertailla ja kokeilla markkinoilta löytyviä Unity-yhteensopivia moninpelipalvelimia. Unity ja sen käyttäminen olivat tuttuja jo ennen työn aloittamista, mutta alustan moninpeliominaisuuksiin en ollut kovin laajasti tutustunut.

Päädyin toteuttamaan moninpelisovelluksen ensin ja paneutumaan palvelinratkaisuihin sen jälkeen. Päätös lienee ollut oikea, sillä palvelinalustojen vertailuun olisi voinut käyttää helposti useitakin kuukausia. Peliprojektin työmäärä oli helpompi arvioida ja sen laajuus oli paremmin rajattavissa.

Peliprojekti lähti liikkeelle sujuvasti eikä koko sen toteuttamisen aikana kohdattu suuria vaikeuksia. Alkuperäisiä suunnitelmia muokattiin tarkoitukseen paremmin sopiviksi matkan aikana, mutta uskon lopputuloksen palvelevan tarkoitustaan hyvin. Opin projektin aikana paljon uutta ja vahvistin myös olemassaolevaa osaamistani.

Opinnäytetyön tutkimusosuus aloitettiin opiskelukiireiden vuoksi hiukan aikataulusta myöhässä ja se kärsi muutenkin juuri meneillään olevista opinnoista johtuvasta ajan ja henkisten resurssien puutteesta. Tutkimuksen aluksi tehdyt suunnitelmat sekä muistiinpanot mahdollistivat kuitenkin sen saattamisen loppuun kohtuullisessa ajassa. Myös opinnäytetyön aiemmassa vaiheessa saatu kokemus oli avuksi.

Peliprojektin lopputulokseen olen hyvin tyytyväinen. Tutkimuksen osalta vertailua olisi voinut tehdä hieman laajemmin. Olisin mielelläni myös testannut vertailtavia tuotteita enemmän.

LÄHTEET

1. Fantastec Ltd.2014. Fantastec Oy.Saatavissa: http://www.fantastec.fi/company.php. Hakupäivä 23.1.2014

2. Woolley, David R. 1994. PLATO: The Emergence of Online Community.Saatavissa: http://thinkofit.com/plato/dwplato.htm. Hakupäivä 16.12.2013.

3. History of Online Gaming. 2008. UGO. Saatavissa: http://www.ugo.com/games/history-of-online-gaming. Hakupäivä 16.12.2013.

4. Heikkilä, Ville-Matias 2013. Internetit ennen Internetiä: modeemipurkkien nousu ja tuho. Saatavissa: http://skrolli.fi/internetit-ennen-interneti%C3%A4-modeemipurkkien-nousu-ja-tuho. Hakupäivä 3.1.2014.

5. Winterhalter, Ryan 2011. Why Quake Changed Games Forever. Saatavissa: http://www.1up.com/features/why-quake-changed-games-forever. Hakupäivä 3.1.2014.

6. Lehtinen, Lasse 1996. Quake C Released. Saatavissa: http://www.satanicslaughter.com/news/?news_id=94. Hakupäivä 3.1.2014.

7. Hope, Dan. A History of MMORPG: The Dungeons & Dragons Legacy. Saatavissa: http://mmorpg-service-review.toptenreviews.com/a-history-of-mmorpg-the-dungeons-dragons-legacy.html. Hakupäivä 3.1.2014.

8. Holisky, Adam 2010. World of Warcraft reaches 12 million players. Saatavissa: http://wow.joystiq.com/2010/10/07/world-of-warcraft-reaches-12-million-players/. Hakupäivä 3.1.2014.

9. The History of Online Gambling. onlinegambling.com. Saatavissa: http://www.onlinegambling.com/online-gambling-history.htm. Hakupäivä 17.12.2013.

10. Brustein, Joshua 2013. The Profitable Future of Free Mobile Apps. Saatavissa: http://www.businessweek.com/articles/2013-09-19/the-profitable-future-of-free-mobile-apps. Hakupäivä 3.1.2014.

11. Fiedler, Glenn 2010. What every programmer needs to know about game networking. Saatavissa: http://gafferongames.com/networking-for-game-programmers/what-every-programmer-needs-to-know-about-game-networking/. Hakupäivä 28.12.2013.

12. Gambetta, Gabriel. Fast-paced Multiplayer (Part 1): Introduction. Saatavissa: http://www.gabrielgambetta.com/fpm1.html. Hakupäivä 19.12.2013.

13. Fiedler, Glenn 2008. UDP vs. TCP. Saatavissa: http://gafferongames.com/networking-for-game-programmers/udp-vs-tcp/. Hakupäivä 29.12.2013.

14. Boulanger, Jean-Sébastien 2006. Interest Management for Massively Multiplayer Games. Quabéc, McGillUniversity. Saatavissa: http://www.cs.mcgill.ca/~jboula2/thesis.pdf. Hakupäivä 28.12.2013.

15. Hook, Brian 2006. The Quake3 Networking Model. Saatavissa: http://trac.bookofhook.com/bookofhook/trac.cgi/wiki/Quake3Networking. Hakupäivä 28.12.2013.

16. Bernier, Yahn W. 2001. Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization. Saatavissa: https://developer.valvesoftware.com/wiki/Latency_Compensating_Methods_in_Client/Server_In-game_Protocol_Design_and_Optimization. Hakupäivä 19.12.2013.

17. Unity Fast Facts. 2013. Unity Technologies. Saatavissa: http://unity3d.com/company/public-relations. Hakupäivä 16.12.2013.

18. Itterheim, Steffen 2013. Measuring Game Engine Popularity. Saatavissa: http://www.learn-cocos2d.com/2013/07/measuring-game-engine-popularity/. Hakupäivä 16.12.2013.

19. Unity Asset Workflow. 2013. Unity Technologies. Saatavissa: http://unity3d.com/unity/workflow/asset-workflow. Hakupäivä 16.12.2013.

20. The GameObject-Component Relationship.2013. Unity Technologies.Saatavissa: http://docs.unity3d.com/Documentation/Manual/TheGameObject-ComponentRelationship.html. Hakupäivä: 3.1.2014.

21. Deactivating GameObjects.2013. Unity Technologies. Saatavissa: http://docs.unity3d.com/Documentation/Manual/DeactivatingGameObjects.html. Hakupäivä: 3.1.2014.

22. Creating And Using Scripts.2013. Unity Technologies. Saatavissa: http://docs.unity3d.com/Documentation/Manual/CreatingAndUsingScripts.html. Hakupäivä: 3.1.2014.

23. Event Functions.2013. Unity Technologies. Saatavissa: http://docs.unity3d.com/Documentation/Manual/EventFunctions.html. Hakupäivä: 3.1.2014.

24. Creating Scenes.2013. Unity Technologies. Saatavissa: http://docs.unity3d.com/Documentation/Manual/CreatingScenes.html. Hakupäivä: 3.1.2014.

25. Application.LoadLevel.2013. Unity Technologies. Saatavissa: http://docs.unity3d.com/Documentation/ScriptReference/Application.LoadLevel.html. Hakupäivä: 3.1.2014.

26. Prefabs.2013. Unity Technologies. Saatavissa: http://docs.unity3d.com/Documentation/Manual/Prefabs.html. Hakupäivä: 3.1.2014.

27. The Unity Editor.2013. Unity Technologies. Saatavissa: http://unity3d.com/unity/workflow/integrated-editor. Hakupäivä: 3.1.2014.

28. Unity Scripting.2013. Unity Technologies. Saatavissa: http://unity3d.com/unity/workflow/scripting. Hakupäivä: 3.1.2014.

29. State Synchronization Details.2013. Unity Technologies. Saatavissa: http://docs.unity3d.com/Documentation/Components/net-StateSynchronization.html. Hakupäivä: 3.1.2014.

30. Networking Elements in Unity.2013. Unity Technologies. Saatavissa: http://docs.unity3d.com/Documentation/Components/net-UnityNetworkElements.html. Hakupäivä: 3.1.2014.

31. Network.useProxy.2013. Unity Technologies. Saatavissa: http://docs.unity3d.com/Documentation/ScriptReference/Network-useProxy.html. Hakupäivä: 3.1.2014.

32. Network Views. 2013. Unity Technologies. Saatavissa: http://docs.unity3d.com/Documentation/Components/net-NetworkView.html. Hakupäivä: 3.1.2014.

33. RPC Details. 2013. Unity Technologies. Saatavissa: http://docs.unity3d.com/Documentation/Components/net-RPCDetails.html. Hakupäivä: 3.1.2014.

34. Network.OnSerializeNetworkView. 2013. Unity Technologies. Saatavissa: http://docs.unity3d.com/Documentation/ScriptReference/Network.OnSerializeNetworkView.html. Hakupäivä: 3.1.2014.

35. The Alljoyn Open Source Project.2013. AllSeen Alliance. Saatavissa: https://allseenalliance.org/developer-resources/alljoyn-open-source-project. Hakupäivä 4.1.2014.

36. Download page for ElectroServer5.2013.ElectroTank. Saatavissa: http://www.electrotank.com/resources/downloads.html. Hakupäivä 30.12.2013.

37. ElectroTankUniversePlatform.2013.ElectroTank. Saatavissa: http://www.electrotank.com/eup.html. Hakupäivä 30.12.2013.

38. ElectroServer 5.2013.ElectroTank. Saatavissa: http://www.electrotank.com/es5.html. Hakupäivä 30.12.2013.

39. High 5 Games Acquires Electrotank Assets to Strengthen Social Gaming Products. 2013. High 5 Games. Saatavissa: http://www.high5games.com/news/high-5-games-acquires-electrotank-assets-to-strengthen-social-gaming-products. Hakupäivä 30.12.2013.

40. Carrigan, Teresa 2013. ElectroServer 5.4.1 released! Saatavissa: http://www.electrotank.com/forums/showthread.php?13117-ElectroServer-5.4.1-released!.Hakupäivä 30.12.2013.

41. The Frost Cloud Solutions: Game Discovery. 2013.Frost Services Inc. Saatavissa: http://www.frost.io/solutions. Hakupäivä 30.12.2013.

42. Next Generation Cloud Gaming Technology.2013.Frost Services Inc. Saatavissa: http://www.frost.io/about. Hakupäivä 30.12.2013.

43. FrostNet Documentation. 2013.Frost Services Inc. Saatavissa: http://www.frost.io/doc/frostnet/index.html. Hakupäivä 30.12.2013.

44. HTML5 Standard W3C Candidate Recommendation.2013. W3C. Saatavissa: http://www.w3.org/TR/2013/CR-html5-20130806/. Hakupäivä 3.1.2014.

45. FrostNet Roadmap.2013.Frost Services Inc. Saatavissa: https://www.frost.io/frostnet/roadmap. Hakupäivä 30.12.2013.

46. Photon Server Features. 2013. Exit Games. Saatavissa: https://www.exitgames.com/en/OnPremise. Hakupäivä 30.12.2013.

47. Understanding Photon Cloud.2013. Exit Games. Saatavissa: http://doc.exitgames.com/photon-cloud/. Hakupäivä 4.1.2014.

48. Photon Server Subscription Plan Pricing.2013. Exit Games. Saatavissa: https://www.exitgames.com/en/OnPremise/Pricing. Hakupäivä 4.1.2014.

49. Understanding Photon Server. 2013. Exit Games. Saatavissa: http://doc.exitgames.com/photon-server/. Hakupäivä 4.1.2014.

50. Photon 3.0 RC9 Change History.2013. Exit Games. Saatavissa: http://doc.exitgames.com/en/photon-server/Photon30RC9. Hakupäivä 5.1.2014.

51. Player.IO. 2013. Yahoo! Inc. Saatavissa: http://playerio.com/. Hakupäivä 30.12.2013.

52. QuickConnect.2013. Yahoo! Inc. Saatavissa: http://playerio.com/features/quickconnect/. Hakupäivä 5.1.2014.

53. PayVault.2013. Yahoo! Inc. Saatavissa: http://playerio.com/features/payvault/. Hakupäivä 5.1.2014.

54. Player.IO Forum: News and Updates. Saatavissa: http://playerio.com/forum/viewforum.php?f=9. Hakupäivä 5.1.2014.

55. Jobs. 2013. Yahoo! Inc. Saatavissa: http://playerio.com/jobs/. Hakupäivä 5.1.2014.

56. uLink. 2013. MuchDifferent. Saatavissa: http://muchdifferent.com/?page=game-unitypark-products-ulink. Hakupäivä 5.1.2014.

57. Server Setup.2013. MuchDifferent. Saatavissa: http://muchdifferent.com/?page=game-unitypark-architecture-fps. Hakupäivä 5.1.2014.

58. Products. 2013. MuchDifferent. Saatavissa: http://muchdifferent.com/?page=game-buy. Hakupäivä 5.1.2013.

59. Man vs. Machine.2013. MuchDifferent. Saatavissa: http://www.muchdifferent.com/?page=game-world-record. Hakupäivä 5.1.2014.

60. API Reference.2013. MuchDifferent. Saatavissa: http://developer.muchdifferent.com/prevdevsite/api/ulink-1.5.1/. Hakupäivä 5.1.2014.

61. uLink Manual. 2013. MuchDifferent. Saatavissa: http://developer.muchdifferent.com/unitypark/uLink/uLink. Hakupäivä 5.1.2014.

Tuote	FrostNet	UnityPark Suite	Photon Server	Player.IO	Electroserver 5
Kehittäjä/palveluntarjoaja	Frost.IO	MuchDifferent	Exit Games	Yahoo! (ent. PlayerScale)	Electrotank
URL	https://www.frost.io/frostnet	http://www.muchdifferent.com/?page=game-unitypark	https://www.exitgames.com/en/OnPremise	http://playerio.com/	http://www.electrotank.com/es5.html
Palvelun/tuotteen tyyppi	Palvelinohjelmisto	Palvelinohjelmisto	Palvelinohjelmisto	Pilvipalvelu	Palvelinohjelmisto
Tuetut palvelinkäyttöjärjestelmät	Windows, Linux, OSX	Windows, Linux, OSX	Windows	-	Windows, Linux, OSX
Ilmainen / kokeiluversio saatavilla?	On	On	On	On	On
Hostauspalvelu saatavilla?	On (cloud.frost.io)	Ei	Ei	On (kuuluu tuotteeseen)	Ei
Kohdemarkkinat / mihin tarkoitettu	Reaaliaikaiset verkkopelit	Kaikenlaiset verkkopelit. MMO:t mainittu erikseen.	Kaikki verkkopelit	Flash- ja Unity-pohjaiset moninpelit	Kaikki verkkopelit, saatavilla myös MMO-kehitykseen tarkoitettu tuotepaketti
Lisenssityypit ja hinnat	Ei tiedossa.	Indie-lisenssi 550EUR / julkaistu tuote, enterprise-lisenssit neuvoteltavissa erikseen	Sovellus-/palvelinkohtaiset lisenssit: <= 100 CCU: ilmainen <= 500 CCU: $500 > 500 CCU: $3500 Tilauslisenssit (per sovellus/serveri): <= 100 CCU: ilmainen <= 500 CCU: $25 / kk > 500 CCU: $175 / kk Enterprise (rajattomasti sovelluksia/servereitä): $750/kk tai $1500/kk	<= 500 CCU: Ilmainen <= 5000 CCU: 24,95$/kk <= 25000 CCU: 500$/kk > 25000 CCU: Neuvoteltava erikseen Lisensseissä myös rajoitukset dataliikenteelle, tietokantaobjektien määrälle sekä levytilalle	<= 50 CCU: Ilmainen <= 1000 CCU: $999 > 1000 CCU: $4999
Referenssejä	Ei ole, uusi tuote	Battlestar Galactica Online, Shadowgun Deadzone	Guns of Icarus, World Golf Tour, King's Bounty: Legions	Ei tiedossa	Ei tiedossa
Palvelimen tukemat skriptauskielet	.NET	Unityn tukemat kielet	.NET	.NET	Java, Python
Verkkoprotokollat	TCP, UDP, WebSockets	UDP	TCP, UDP, HTTP	TCP	TCP, UDP, HTTP
Interest management	On	On	On	Ei	On
Verkkoliikenteen salaus	On	On	On	On	On

Tuote	FrostNet	UnityPark Suite	Photon Server	Player.IO	Electroserver 5
Automaattinen palvelininstanssien hallinta	Ei	Saatavilla erikseen	Ei	On	Ei (1)
Kuormantasaus palvelinten välillä	Tulossa	Saatavilla erikseen	On	On	Ei (1)
Viestijärjestelmä (viestikanavat, liittyminen/poistuminen, private chat)	Ei	Ei. Viestijärjestelmä täytyy toteuttaa erikseen, mutta kanavointi voidaan tehdä uLinkin toiminnoilla	Ei	On	On
Tiedonsiirtomenetelmät	Event-systeemi	RPC, tilasynkronointi	RPC, tilasynkronointi	Dataviestit, yhdessä viestissä voidaan lähettää useita erilaisia datatyyppejä	Request-/response-viestit ja eventit
Web UI for server administration	Ei	Ei	Dashboard statistiikkojen katseluun	On	On
Sisältääkö tietokantapalvelimen?	Ei	On (no-sql)	Ei	On (no-sql)	Ei (tukee ODBC/JDBC-yhteyksiä ulkoisiin tietokantoihin
Tuki ulkoisille autentikointipalveluille?	Ei	Ei	Ei	On (Facebook, Kongregate)	Ei
Streamauspalvelin	Tulossa	Ei	Ei	Ei	On
Lobbytoiminnallisuus?	Tulossa	On	On	Esimerkkitoteutus	On
Dokumentaation laatu ja kattavuus	Keskeneräinen	Ok	Ok	Dokumentaatio OK, Unity-esimerkki rikki (Unity-versiossa 4.3.1)	Ok
Onko esimerkkiprojekteja?	On	On	On	On	On
Julkinen roadmap?	On	Ei	Ei	Ei	Ei
Rajapinnat ylläpitoa varten	Komentorivi	Ei erillisiä rajapinto-ja, palvelinsovellukset Unity-ohjelmia → Käytössä Unityllä toteutetut ylläpitotoiminnot	WWW, telnet, Windows-sovellus	WWW	WWW

Tuote	FrostNet	UnityPark Suite	Photon Server	Player.IO	Electroserver 5
Unity-integraatio	Hyvä. FrostNetin skriptit on tehty helppokäyttöisiksi ja esimerkiksi verkkoyhteyden omiaisuuksia ja simulointia voidaan säätää editorista. Palvelimen käyttävä kirjastomoduuli käännetään osin samoista kooditiedostoista kuin asiakasohjelma.	Hyvä, uLinkin omat komponentit löytyvät komponenttivalikosta ja ovat helppokäyttöisiä.	Ok. Itse kirjasto ei toteuta editoriin toiminnallisuutta, mutta esimerkkiprojekteista löytyvät komponentit ovat hyvin toteutettuja ja melko käyttökelpoisia sellaisenaankin.	Heikohko. Editoriin integroituja toimintoja ei ole.	Ei selvillä, tätä tuotetta ei kokeiltu käytännössä.
Muuta	Käyttää Lidgren-verkkokirjastoa. Dokumentaatio pahasti keskeneräinen, tarkempaa testaamista voidaan tehdä jos aikaa jää valmiimmilta tuotteilta. Ensimmäinen ja viimeisin SDK:n julkaisu tehty 26.8.2013.	Sekä clientti että palvelin ovat Unity-sovelluksia.	Photon Cloud (Realtime) on tuotteen pilvipalveluversio, jossa ei ainakaan toistaiseksi ole mahdollisuutta oman koodin ajamiseen palvelimella.	Tuote on tarkoitettu kevyiden selain-/mobiilipelien palvelimeksi.	High 5 Games osti Electrotankin kesäkuussa 2013, tuotteen nykytilasta ei ole tietoa. Viimeisin päivitys tuotteeseen tehty keväällä 2013, mutta toisaalta kehittäjät tukevat tuotetta edelleen aktiivisesti.