KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

Toni Paavola

TYÖKALUT ETÄTYÖNÄ TOTEUTETUSSA ANDROID -PELIKEHITYKSESSÄ

Opinnäytetyö
Maaliskuu 2014

POHJOIS-KARJALAN
AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Maaliskuu 2014
Tietojenkäsittelyn koulutusohjelma

Karjalankatu 3
80200 JOENSUU
p. (013) 260 600

Tekijä(t)
Toni Paavola

Nimeke
Työkalut etätyönä toteutetussa Android-pelikehityksessä

Toimeksiantaja
-

Tiivistelmä

Tässä opinnäytetyössä käydään läpi työkaluja, joita tarvitaan etätyönä tapahtuvassa mobiilipelikehityksessä. Raportissa tarkastellaan, mihin tarkoituksiin työkaluja tarvitaan etätyönä tehtävässä Android-pelikehityksessä ja miten ne valikoituivat käytettäväksi tässä projektissa.

Opinnäytetyössä selvitetään, miten valitut työkalut soveltuvat käytettäväksi etätyöskentelyssä, miten ne soveltuvat Android-pelikehitykseen ja mitä hyviä ja huonoja puolia niillä on etätyöskentelyn ja Android-pelikehityksen kannalta. Työkalujen lisäksi raportissa tutustutaan yleisesti muun muassa Android-mobiilikäyttöjärjestelmään, etätyöhön, pariohjelmointiin ja versionhallintaan. Toiminnallisessa osuudessa tutustutaan hieman itse pelin prototyypin tekemiseen. Pääpaino on kuitenkin käytetyissä työkaluissa.

Opinnäytetyön tuloksena selvisi, että moobiilipelikehitys on mahdollista toteuttaa etätyöskentelyn avulla. Projektiin valitut ohjelmat toimivat yleisesti ottaen hyvin sekä etätyössä, että pelikehityksessä. Toiminnallisessa osuudessa syntyi tower defence -pelin prototyyppi.

Kieli
suomi

Sivuja
52

Asiasanat
Android, mobiilipelikehitys, etätyö, työkalut mobiilikehityksessä

NORTH KARELIA
UNIVERSITY OF APPLIED SCIENCES

THESIS
March 2014
Degree Programme in Business Information Technology

Karjalankatu 3
80200 JOENSUU
FINLAND
Tel. 358-13-260 600

Author(s)
Toni Paavola

Title
Tools in Remote Android Game Development

Commissioned by
-

Abstract

This thesis focuses on tools that are needed in mobile game development that is done with remote work. This report will go through for what purposes the tools are needed in remote work Android game development and how the tools were chosen for this project.

This thesis pursues to answer the following research questions: how well the chosen tools suit remote work, how they lend themselves to Android game development, and what advantages or disadvantages they have when it comes to remote work and Android game development. In addition to the tools this report gives a brief introduction to Android mobile operating system, remote work, pair programming and version control. Making of the prototype game is explained in the practical part of the thesis but the main aspect is still in the used tools.

The project proved that mobile game development is doable with remote work. The tools selected to this project worked generally well in both remote work and mobile game development. A prototype of a tower defence game was made in the practical part of the thesis.

Language
Finnish

Pages
52

Keywords
Android, mobile game development, remote work, tools in mobile development

Sisältö

1 Johdanto

Tässä opinnäytetyössä käsittelen työkaluja, joita tarvitaan Android-pelikehityksessä. Käytännön osuudessa kerron, kuinka toteutimme yhdessä Heikki Ylösen kanssa rungon tower defence -tyyppiselle pelille Android-laitteille. Tower defence -pelien ideana on estää, yleensä useammassa syklissä ilmestyvien, vihollisjoukkojen pääsy pisteestä A pisteeseen B. Estäminen tapahtuu erilaisten puolustustornien avulla, joita rakennetaan ampumaan tai muuten estämään vihollisjoukon eteneminen.

Pelaaminen mobiililaitteilla, kuten kännyköillä ja tablet-tietokoneilla on nykyään hyvin yleistä. Nykyään kaikista ladatuista mobiilisovelluksista 33 % on pelejä, ja kaikesta mobiilisovellus tuotosta pelien osuus on 66 % (Newzoo 2013). Mobiilipelaaminen on isoa bisnestä ja sitä tehdään paljon myös Suomessa. Hyvinä esimerkkeinä tästä ovat esimerkiksi suomalaiset pelitalot Rovio ja Supercell. Mobiilipelaamisen nykytilanteen johdosta halusimme perehtyä mobiilipelien kehittämiseen tarkemmin tämän projektin myötä.

Projektillamme ei ole toimeksiantajaa, vaan se on minun ja Heikin ideoima ja toteuttama. Opinnäytetyön osalta tarkoituksena oli toteuttaa tower defence -pelin runko. Projektin lähdekoodi tulee olemaan täysin vapaasti ladattavissa, joten kuka tahansa voi halutessaan tutustua siihen ja kehittää sitä eteenpäin haluamallaan tavalla. Ajattelimme myös mahdollisesti itse jatkaa pelin kehitystä tulevaisuudessa. Tämä projekti kuitenkin keskittyy enemmän mobiilikehitykseen tutustumiseen.

Oma teoriaosuuteni keskittyy erilaisiin työkaluihin, joita mahdollisesti tarvitaan sovellusten, ja erityisesti pelien, kehittämisessä mobiilialustoille. Keskityn lähinnä niihin työkaluihin, joita käytimme itse tämän projektin toteutuksessa. Kerron varsinaisista ohjelmointityökaluista kuten Eclipse-kehitysympäristöstä, mutta myös esimerkiksi yhteydenpitoon käyttämistämme ohjelmista kuten Adobe Connect-verkkokokoussovelluksesta. Yritän tarkastella mahdollisuuksien mu-

kaan ohjelmien käyttöä etätyöskentelyn näkökulmasta ja niiden soveltuvuutta siihen. Otin tämän näkökulman siksi, että toteutimme projektityöskentelyn fyysisesti eri paikkakunnilta. Heikki työskenteli suurimman osan ajasta Helsingistä käsin ja itse olin Joensuussa. Kehitysmenetelmänä käytimme suurimmaksi osin pariohjelmointia, mutta teimme töitä myös itsenäisesti. Tämä opinnäytetyö pyrkii vastaamaan sovellusten osalta seuraaviin tutkimuskysymyksiin: miten valitut ohjelmat soveltuvat käytettäväksi etätyöskentelyssä, miten ne soveltuvat Android-pelikehitykseen ja mitä hyviä ja huonoja puolia niillä on etätyöskentelyn ja Android-pelikehityksen kannalta.

2 Projektin toteuttamisen edellytykset

Opinnäytetyöprojektin lopputuloksena oli siis tarkoitus syntyä tower defence -pelin prototyyppi tai perusrunko Android-mobiilialustalle. Alustavan suunnitelman mukaan prototyyppiin tulisi aloitusruutu, kun pelin käynnistää. Aloitusruudussa näkyisi pelin nimi sekä minun ja Heikin nimet. Aloitusruudun jälkeen kuva siirtyisi päävalikkoon, mistä voi valita mitä kenttää haluaa pelata. Kentän valinnan jälkeen peli käynnistyisi ja kentän läpäisemisen tai epäonnistumisen jälkeen tulisi lopetusruutu, minkä jälkeen näytettäisiin taas päävalikko. Emme halunneet laittaa pelin prototyyppiin mitään ääniä, sillä siinä olisi kulunut turhaa aikaa. Grafiikat päätimme tehdä itse tai teettää jollakin osaavammalla taholla.

Jotta projekti saataisiin vietyä loppuun, meidän täytyisi käyttää apuna erilaisia ohjelmia. Koska asuimme eri paikkakunnilla, etätyöskentelystä aiheutuvat ongelmat pitäisi saada ratkaistua. Päätimme käyttää kehitysmenetelmänä pariohjelmointia, joka on perinteisesti tarkoittanut sitä, että kaksi ihmistä työskentelee yhdessä yhden tietokoneen äärellä. Tätä toimintamallia pitäisi pystyä nyt soveltamaan etätyöskentelyssä.

Etätyöskentelyn ja pariohjelmoinnin yhteensovittamiseen tarvitaan ohjelma tai ohjelmia, joilla pystytään kommunikoimaan äänen avulla. Koska pariohjelmoin-

nin keskeinen toimintamalli on se, että toinen katsoo vierestä mitä toinen tekee, tarvitaan myös jokin ohjelma oman työpöytänäkymän jakamiseksi työparille. Ohjelma, jolla työpöydän voi antaa toisen hallittavaksi, olisi myös varmasti hyödyllinen.

Etätyöskentelystä johtuen meidän täytyisi jotenkin huolehtia siitä, että kummallakin on koko ajan uusimmat tiedostot ja muutokset saatavilla. Jokaisessa ohjelmistoprojektissa on hyvä käyttää jonkinlaista versionhallintaa. Versionhallinnan avulla pystytään seuraamaan projektiin tehtyjä muutoksia ja siirtymään eri versioista toisiin. Versionhallinta mahdollistaa myös eri tiedostojen samanaikaisen muokkaamisen. Versionhallinnan tärkeys korostuu etätyöskentelyssä, sillä projektin jäsenten täytyy pystyä muokkaamaan tiedostoja ilman, että täytyy pelätä ylikirjoittavansa muiden tekemiä muutoksia. Voisimme tietysti aina ilmoittaa toisillemme, kun alamme muokkaamaan jotain tiedostoa ja muokkaamisen jälkeen lähettää tiedostot toisillemme, mutta versionhallinnan avulla tämä kaikki hoituu paljon helpommin.

Kun etätyöskentelystä johtuvat ongelmat saataisiin ratkaistua, pitäisi miettiä miten itse ohjelmointityö ja muu pelin kehittäminen hoidettaisiin. Tarvitsisimme jonkin kehitysympäristön, jonka sisällä kaikki tapahtuisi. Kehitysympäristön lisäksi tarvitsisimme pelille jonkinlaisen alustan eli pelimoottorin, jonka avulla pelin tekeminen olisi huomattavasti helpompaa, kuin jos tekisimme kaiken alusta asti itse. Tarvitsisimme myös ohjelman, jolla peliin tulevat kentät saataisiin tehtyä.

3 Android-mobiilikäyttöjärjestelmä projektin alustana

Valitsimme projektimme toteutusalustaksi Androidin pääosin siksi, että se on hyvin yleinen mobiilikäyttöjärjestelmä. Sekä minä että Heikki omistimme entuudestaan Android-laitteen, mikä lisäsi mielenkiintoa projektia kohtaan. Pystyimme myös hyödyntämään laitteitamme pelin testauksessa. Arvostamme molem-

mat myös sitä, että Android perustuu vapaaseen lähdekoodiin. Itselleni Android-ohjelmointi oli entuudestaan hieman tuttua, sillä olin suorittanut Android-ohjelmoinnin peruskurssin. Android-kehityksen aloittaminen on melko helppoa, sillä kehitykseen on olemassa kattavat työkalut.

3.1 Yleistä Androidista

Android on käyttöjärjestelmä mobiililaitteille. Sen ovat alun perin kehittäneet Andy Rubin, Rich Miner, Nick Sears sekä Chris White, jotka perustivat Android-yhtymän vuonna 2003. Vuonna 2005 Android-yhtymä siirtyi kokonaan Googlen omistukseen. (EngineersGarage 2012.)

Android perustuu avoimeen lähdekoodiin, mikä tarkoittaa sitä, että sen käyttäminen ja sovellusten kehittäminen sille on ilmaista. Android eroaa esimerkiksi Applen iOS käyttöjärjestelmästä siten, että monet puhelinvalmistajat käyttävät Androidia, siinä missä iOS löytyy vain Applen tuotteista. Androidia käytetään toki myös muissakin mobiililaitteissa, kuin vain puhelimissa. Esimerkiksi monissa tablet-tietokoneissa pyörii Android. Suurimpia Android käyttöjärjestelmää käyttäviä laitevalmistajia ovat: HTC, LG, Samsung, Motorola ja Sony.(Androidsuomi.fi 2014.)

Yleisimmin sovellukset Androidille hankitaan Google Play Storesta. Se on Googlen ylläpitämä virallinen sovelluskauppa verkossa ja sen kautta voidaan ostaa ja ladata sovelluksia laitteille. Vaihtoehtoisia sovelluskauppoja on esimerkiksi Amazonilla. Jotkut sovelluksia tekevät tahot mahdollistavat sovellusten hankkimisen myös suoraan valmistajalta esimerkiksi heidän nettisivuiltaan. Sovelluksia on sekä ilmaisia että maksullisia ja niitä on saatavilla valtava määrä. Heinäkuussa 2013 sovellusten määrä Googlen Play Storessa ylitti miljoonan rajan (Victor 2013). Kuukausittaisia latauksia kaupassa suoritetaan 1.5 miljardia. Play Store on avoin sovelluskauppa, mikä tarkoittaa sitä, että kuka tahansa voi kehittää sovelluksen ja jakaa sen muiden käyttäjien saataville. (Androidsuomi.fi 2014.)

3.2 Android-laitteet

Samsung on suurin Androidia käyttävä laitevalmistaja. Sen osuus kaikista Android-laitteista on 47.5 %. Toisena huomattavasti pienemmällä markkinaosuudella on Sony-Ericsson 6.5 % osuudellaan. Muita pienempiä valmistajia ovat mm. Motorola, HTC ja LG. (OpenSignal 2013.)

3.2.1 Älypuhelimet

Älypuhelimella tarkoitetaan matkapuhelinta, joka lisää puhelimeen kämmentietokoneille tyypillisiä toiminnallisuuksia ja ominaisuuksia. Älypuhelinten käyttöjärjestelmät mahdollistavat erilaisten sovellusten käyttämisen laitteella. Näitä sovelluksia ovat esimerkiksi tekstinkäsittelyohjelmat, sähköpostiohjelmat ja erilaiset pelit. Älypuhelimilla on yleensä mahdollista luoda internet-yhteys ja selata internet-sivuja. Nykyään useimmissa älypuhelimissa on graafiset käyttöliittymät ja kosketusnäytöt. (Cassavoy 2013).

Nykyajan älypuhelin yhdistää monien eri laitteiden ominaisuuksia. Itselläni älypuhelin on korvannut perinteisistä laitteista kokonaan mm. kameran ja mp3-soittimen. Puhelimella hoidan myös usein pankkiasioita sekä luen sähköpostit.

3.2.2 Tablet-tietokoneet

Tablet-tietokonetta voisi kuvailla älypuhelimen ja kannettavan tietokoneen risteytykseksi. Tableteissa ei yleensä ole fyysistä näppäimistöä, vaan laitteen käyttäminen tapahtuu kosketusnäytön avulla, kuten monissa älypuhelimissa. Jossain hybridilaitteissa on saatavilla myös näppäimistö. Tablet-tietokoneiden näyttöjen koot vaihtelevat yleensä 7 ja 13 tuuman välillä. (Tablet-PC 2014.)

Ensimmäisen tablet-tietokoneen kehitti Apple jo vuonna 1989, mutta tablettien yleistyminen alkoi vasta vuonna 2010 Apple iPadin myötä. Applen iPadien lisäksi suosituimpiin tabletteihin kuuluu Samsungin Galaxy Tab-laitteet, joissa on Android-käyttöjärjestelmä. Normaalin hyötykäytön lisäksi tabletteja käytetään paljon pelaamiseen. (Tablet-PC 2014.)

3.3 Sovellusten kehittäminen Androidille

Android-puhelimet olivat maailman eniten myydyimpiä puhelimia vuonna 2013, jolloin niiden markkinaosuus oli 79 %. Myynnin kasvu kuitenkin hidastui verrattuna aikaisempiin vuosiin. Vaikka kasvu hidastuikin, Android on silti myydyin käyttöjärjestelmä puhelimissa ja tästä syystä sovellustarjonta Androidille on laaja ja sovelluskehitys Androidille mahdollistaa suuren käyttäjäkunnan tavoittamisen. Muita laajalti käytössä olevia mobiilikäyttöjärjestelmiä ovat Applen iOS, BlackBerry OS, Windows sekä Linux. (Lomas 2014.)

Android-alustalle on olemassa valmiit työkalut sovellusten kehittämiseen. Eclipse-kehitysympäristöön (kts. 5.2) on saatavilla Android Developer Tools (kts.5.3) lisäosa, joka mahdollistaa sovellusten kehittämisen Android-alustalle. ADT on ilmainen ja perustuu avoimeen lähdekoodiin. Se tarjoaa erittäin hyvät työkalut sovellusten tekemiseen ja testaamiseen. Myös Eclipse on ilmainen kehitysympäristö, joten sovelluskehitys Androidille on täysin ilmaista. Itse ohjelmointi tapahtuu Java ohjelmointikielellä. (Developer Android 2014.)

Android-laitteiden, sekä itse käyttöjärjestelmän pirstaloituneisuus on eräs Androidin ongelma. Laitteita on valtava määrä erilaisia ja ne eroavat ominaisuuksiltaan huomattavasti toisistaan. Vuonna 2013 erilaisia Android-laitteita on markkinoilla lähes 12000 erilaista ja eri Android-versioita on käytössä kahdeksan (taulukko 1). (OpenSignal 2013.) Kirjoitushetkellä on jo julkaistu uusin versio 4.4, joka kantaa nimeä KitKat (Android 2014).

Android-versio	Nimi
4.1-4.3	Jelly Bean
4.0	Ice Cream Sandwich
3.X	Honeycomb
2.3	Gingerbread
2.2	Froyo
2.0/2.01/2.1	Eclair
1.6	Donut
1.5	Cupcake

Taulukko 1. Kuvaus käytössä olevista Android-versioista (Hildenbrand 2013).

Sirpaleisuus aiheuttaa kehittäjille ongelmia, sillä sovellusten kehittämisessä täytyy ottaa huomioon eri laitteiden ja Android-versioiden erot. Jos halutaan tavoittaa mahdollisimman suuri käyttäjäkunta, voi erojen huomioonottamisen johdosta kehitys olla hankalaa ja aikaa vievää. Sirpaleisuudessa on myös hyviä puolia. Vanhemmat laitteet, jotka toimivat Androidin aikaisemmilla versioilla, ovat melko halpoja ja niitä on paljon. Tästä syystä käyttäjäkunta on laaja ja mahdollisia asiakkaita on paljon. Käyttäjälle tämä antaa mahdollisuuden hankkia juuri sellainen laite, kuin itselle sopii. Laitteita on monen hintaisia ja ominaisuuksista löytyy varmasti itselle mieleinen. (OpenSignal 2013.)

4 Etätyöskentely työskentelymetodina

Etätyöskentely on työskentelyn malli, jossa töitä tehdään jossain muualla, kuin varsinaisella työpaikalla. Töitä tehdään usein kotoa, tai esimerkiksi matkoilta käsin. Etätyöksi katsotaan myös työnantajan eri toimipisteissä, työkohteissa tai asiakkaan luota tapahtuva työskentely. Etätyöskentely mahdollistaa joustavamman työskentelemisen, kuin perinteinen fyysisesti työpaikalla tapahtuva työnteko. (Työ- ja elinkeinoministeriö 2008.)

Kotoa tapahtuva työskentely on kasvanut 1990-luvun alusta lähtien vuoteen 2008 saakka, jolloin noin 15 % suomalaisista teki osan töistä kotonaan. Tällä hetkellä työterveyslaitoksen tutkimuksen mukaan luku on 14 %, joten kasvu näyttää pysähtyneen. (Hakala 2013.)

Etätyöllä saavutetaan monia hyötyjä, mutta siitä saattaa olla myös haittaa. Suurin yksittäinen hyöty on työajan joustavuus. Töitä voidaan tehdä silloin kun se itselle parhaiten sopii ja esimerkiksi työn ja perhe-elämän yhteensovittaminen helpottuu. Myös työmatkoihin kuluva aika säästyy, jolloin säästynyt aika voidaan käyttää johonkin muuhun. (Työ- ja elinkeinoministeriö 2008.)

Varsinkin kotoa käsin tapahtuvassa työskentelyssä on myös monia negatiivisia puolia. Työ ja vapaa-aika saattavat sekoittua ja työaika saattaa venyä. Etätyöskentelijä on myös usein yksin ja esimerkiksi toimistotyöskentelyn sosiaalinen kanssakäyminen jää puuttumaan kokonaan. Kotona ei myöskään välttämättä ole kaikkia tarvittavia välineitä tehokkaaseen työskentelyyn. (Saavalainen 2013.)

Etätyöskentely sopii erityisen hyvin it-alalla työskenteleville, jotka tarvitsevat ainoastaan tietokonetta työnsä tekemiseen. Etätyö antaa paljon vapauksia, mutta vaatii myös tekijältä aktiivisuutta ja itsekuria. Kotona on helppo unohtaa, että töitä pitäisi todella tehdä ja etätyössä motivaatio on helpompi hukata, kuin fyysisesti työpaikalla töitä tehdessä. Kokoukset, palaverit ja muut tapahtumat, joihin yleensä kokoonnutaan työporukalla, voidaan hoitaa etänä, mutta näistä tapahtumista jää kuitenkin puuttumaan tietty tunnelma, ja sosiaalinen puoli työnteossa kärsii.

4.1 Pariohjelmointi kehitysmenetelmä

Pariohjelmointi kuuluu ketteriin kehitysmenetelmiin. Siinä kaksi ohjelmoijaa työskentelee yhdessä yhden työpisteen äärellä tehden samaa työtehtävää. Toinen työntekijöistä tekee varsinaisen ohjelmoinnin ja toinen katsoo vierestä ja

antaa neuvoja ja ehdotuksia sekä kyselee. Varsinaista ohjelmointia tekevä työntekijä keskittyy syntaksiin ja matalan tason ohjelmointiin, kun taas vierestä seuraaja miettii työtehtävää laajemmasta näkökulmasta. Työntekijät voivat vaihtaa välillä rooleja. (Versionone 2013.)

Pariohjelmoinnilla pyritään parantamaan ohjelmoinnin laatua, vähentää ohjelmointivirheitä ja lisätä tietoa työntekijöiden keskuudessa. Aalto-yliopistossa tarkistetun väitöskirjan mukaan pariohjelmointi paransi ohjelmistojen laatua ja ohjelmat tulivat paremmin tutuiksi eri ihmisille. Ohjelmointivirheiden vähentyminen johtuu siitä, että suunnitelmat ja toteutus tehdään yhdessä, jolloin työskentely on huolellisempaa. Kun koodia tutkii jatkuvasti kaksi ihmistä, virheet huomataan todennäköisemmin. Pariohjelmointi soveltuu erityisesti työn suunnitteluvaiheeseen, sillä se koetaan yleensä ohjelmoinnin vaikeimmaksi vaiheeksi. (Aalto-yliopisto 2011.)

Tutkimustuloksissa on osoitettu, että lyhyellä aikavälillä, pariohjelmointia käytettäessä, tuottavuus saattaa laskea, varsinkin jos sitä mitataan kirjoitettujen koodirivien määrällä. Koodia syntyy vähemmän, koska vain toinen ohjelmoijista kirjoittaa varsinaista ohjelmakoodia. Pidemmällä aikavälillä tarkasteltuna tuottavuus kuitenkin nousee. Tämä johtuu siitä, että tuotettu koodi on parempaa ja siinä on vähemmän virheitä. Virheiden etsimiseen ja korjaamiseen kuluu vähemmän aikaa. (Versionone 2013.)

Omassa projektityössämme toteutimme pariohjelmointia etätyönä. Käytännössä toteutus on täysin samanlainen, kuin fyysisesti samassa tilassa pariohjelmoiminen, mutta etätyöskentely aiheutti hieman ongelmia, koska emme olleet kasvotusten yhden työpisteen äärellä. Keskustelu ei ole yhtä helppoa ja pitäisi myös pystyä näkemään mitä toinen tekee, jotta voi antaa neuvoja ja kommentoida toisen tekemisiä. Siitä miten nämä ongelmat voidaan ratkaista, kerron enemmän seuraavassa luvussa.

4.2 Etätyön mahdollistamiseksi käytetyt ohjelmat

Kuten jo aiemmin mainitsin, toteutimme projektityöskentelymme suurimmaksi osaksi pariohjelmointina. Pariohjelmoinnin ja etätyöskentelyn yhteensovittamiseen tarvitaan joitakin ohjelmia. Kummankin osapuolen tulee pystyä näkemään sekä kuulemaan, mitä toinen tekee. Onneksi sekä äänen että kuvan välittämiseen on tarjolla hyviä ohjelmia.

4.2.2 Adobe Connect -web-konferenssiohjelma

Adobe Connect on tarkoitettu palaverien, tapaamisten, esittelyjen, konferenssien ym. pitämiseen verkossa. Se toimii verkkoselaimessa, joten varsinaista asiakasohjelmaa ei tarvitse asentaa. Tapaamisiin osallistuja tarvitsee nettiyhteyden, verkkoselaimen ja Adobe Flash Player 10.1 tai uudemman. Jos osallistuja haluaa esimerkiksi jakaa oman työpöytänsä kaikkien nähtäville, tulee hänen asentaa Adobe Connect lisäosa. Sovelluksesta on saatavilla myös mobiiliversio iOS-, Android- ja Blackberry PlayBook-laitteille. Adobe Connect on yksi Adobe Systems:in tuotteista. Adobe Systems:in muita tuotteita ovat mm. Adobe Reader tiedostojenlukuohjelma sekä Adobe Flash Player. (Adobe 2014.)

Adobe Connectin tärkeimmät ominaisuudet ovat kuvan ja äänen jakaminen muille osallistujille. Käyttäjä pystyy jakamaan esimerkiksi oman työpöytänsä kaikkien nähtäville. Ohjelman avulla voi kommunikoida äänellä, sekä esittää esimerkiksi powerpoint- esityksiä. Osallistujat voivat keskustella toisilleen myös kirjoittamalla chat-ikkunassa. Tiedostojen lähettäminen ja lataaminen on mahdollista ja muille osallistujille voi antaa etä-käyttöoikeuden omaan tietokoneeseen. (Adobe 2014.)

4.2.3 Skype-pikaviestiohjelma

Skype on sovellus, jonka avulla on mahdollista kommunikoida ihmisten kanssa verkon välityksellä. Skypellä voi soittaa ja lähettää viestejä toisille ihmisille, jotka ovat myös skypessä. Puheluja voi puhua kaksin tai useamman ihmisen kanssa ryhmäpuhelussa. Matka- ja lankapuhelimiin soittaminen on myös mahdollista. Mikäli laitteessa, jolla Skypeä käytetään, on kamera, onnistuu myös videopuheluiden soittaminen. (Skype 2014.)

Viestejä voi lähettää monella eri tapaa. Skypestä toiseen voi lähettää videoviestejä sekä tekstimuotoisia pikaviestejä. Skypestä matkapuhelimiin voi lähettää normaaleja tekstiviestejä. Ääniviestien jättäminen on mahdollista, kun toinen osapuoli ei pääse vastaamaan puheluun. (Skype 2014.)

Eräs merkittävä ominaisuus on tiedostojen jakaminen. Skypen välityksellä voi lähettää valokuvia, videoita ja muita tiedostoja. Lähettäminen tapahtuu vetämällä tiedosto keskusteluikkunaan. Toinen hyödyllinen jakamisominaisuus on oman näytön jakaminen keskustelun vastapuolelle. Näytön jakamisen jälkeen vastapuoli näkee kaiken mitä omalla ruudulla tapahtuu. (Skype 2014.)

5 Ohjelmoinnissa käytetyt tekniikat ja ohjelmat

Tässä luvussa esitellään projektin tuloksena syntyneen tower defence -pelin rungon tekemiseen käytetyt tekniikat ja työkalut. Pelin kehitys tapahtui Eclipse-kehitysympäristössä, johon oli tuotu AndEngine niminen Android-pelimoottori. Ohjelmointikielenä käytettiin Javaa.

5.1 Java-ohjelmointikieli

Java on ohjelmointikieli ja kehitysalusta, jonka kehitti alun perin Sun Microsystems. Se julkaistiin vuonna 1995 ja useat verkkosivut ja sovellukset vaativat Javan toimiakseen. Java on laajalti käytössä eri laitteissa ja alustoilla kuten tietokoneissa, pelikonsoleissa ja puhelimissa. Java on ilmainen, ja sen voi kuka tahansa ladata ja asentaa tietokoneelleen. (Java 2014.)

Javalla pystytään kehittämään työpöytä- ja mobiilisovelluksia. Myös pelien ja verkkosisällön tekeminen on mahdollista. Java mahdollistaa ohjelmistojen kehittämisen alustariippumattomasti, jolloin tietyllä alustalla kehitetty ohjelmisto toimii myös melkein millä tahansa muulla alustalla. (Java 2014.)

5.2 Eclipse-kehitysympäristö

Eclipse on Java-pohjainen kehitysympäristö. Se pohjautuu avoimeen lähdekoodiin, on ilmainen, ja siihen voidaan liittää rajaton määrä lisäosia. Eclipse sisältää valmiiksi perustyökalut, joihin kuuluu myös Java-kehitykseen tarvittavat JDT-työkalut. Vaikka Eclipse onkin Java-pohjainen, se tukee ohjelmistokehitystä myös muilla kielillä kuten esimerkiksi C ja C++. (Aniszczyk & Gallardo 2007.)

Eclipse tarjoaa ohjelmistokehittäjälle apuvälineitä, joiden avulla kehitystyö helpottuu. Apuvälineitä ovat mm: Syntaksin korostus, joka helpottaa ohjelmakoodin rakenteen hahmottamista, ohjelmakoodin kääntäjä, navigaattori, tiedostohallitsin ja paljon muuta. Eclipse on saatavilla monille eri alustoille kuten Windows, Linux, Solaris ja Mac OS X. (Aniszczyk & Gallardo 2007.)

Eclipseen on saatavilla monia versionhallinta-lisäosia. Versionhallinta tallentaa valittuihin tiedostoihin tehdyt muutokset, jolloin käyttäjä voi palata aikaisempiin versioihin, jos jotain menee vikaan (Git 2014). Yksi Eclipseen saatavista lisäosista on nimeltään MercurialEclipse, jota mahdollistaa Mercurial-versionhallinnan (kts. 6. Versionhallinta) käytön suoraan Eclipsen sisällä. Mer-

curial-versionhallinta täytyy olla asennettuna koneelle, jotta MercurialEclipse-lisäosa toimii. (Mercurial 2014a.)

5.3 Android Developer Tools -kehitystyökalut

Android Developer Tools eli Android-kehitystyökalut (ADT) tarjoavat kaikki työkalut, mitä tarvitaan sovellusten kehittämiseen androidille. Se on lisäosa Eclipse-kehitysympäristöön, joka sisältää mm: Tarvittavat ohjelmakirjastot Android-kehitystä varten, Graafisen käyttöliittymän rakentajan, virtuaalisen Android-laitteen testausta varten sekä mahdollisuuden testata kehitteillä olevaa sovellusta oikeilla Android-laitteilla. (Developer Android 2014.)

ADT:n avulla voidaan luoda Android-projektille nopeasti perusrakenne ja komponentit ohjatun toiminnon avulla. ADT tarjoaa myös paremman muokkaimen Android-sovellusten käyttämille XML-resursseille. XML on merkkauskieli, jonka avulla voidaan siirtää ja säilöä tietoa (w3schools 2014).

5.4 AndEngine-pelimoottori

AndEngine on joukko ohjelmakirjastoja, joiden avulla voidaan tehdä 2D- pelejä Android-alustalle. Ohjelmakirjastolla tarkoitetaan valmiiksi kirjoitettua ohjelmakoodia, joka voidaan lisätä omaan sovellukseen, jottei kaikkea toiminnallisuutta tarvitse tehdä itse alusta alkaen (Techopedia 2014.) AndEngine-kirjastot ovat avointa lähdekoodia, joten kuka tahansa voi käyttää niitä ilmaiseksi. (Ricardo 2013.)

AndEngine tarjoamia ominaisuuksia ovat esimerkiksi automaattinen kuvan skaalaus eri laitteilla, usean kosketuksen tuki kosketusnäytöllä, fysiikkamallinnus sekä tuki Tiled Maps Editorilla luotujen pelikenttien tmx-formaatille (kts 5.5 Tiled Map Editor). Muita hyötyjä ovat sen ilmaisuus ja lähdekoodin avoimuus. Jos tarve vaatii, voi käyttäjä muokata toiminnallisuuksia ja lisätä niitä. AndEngi-

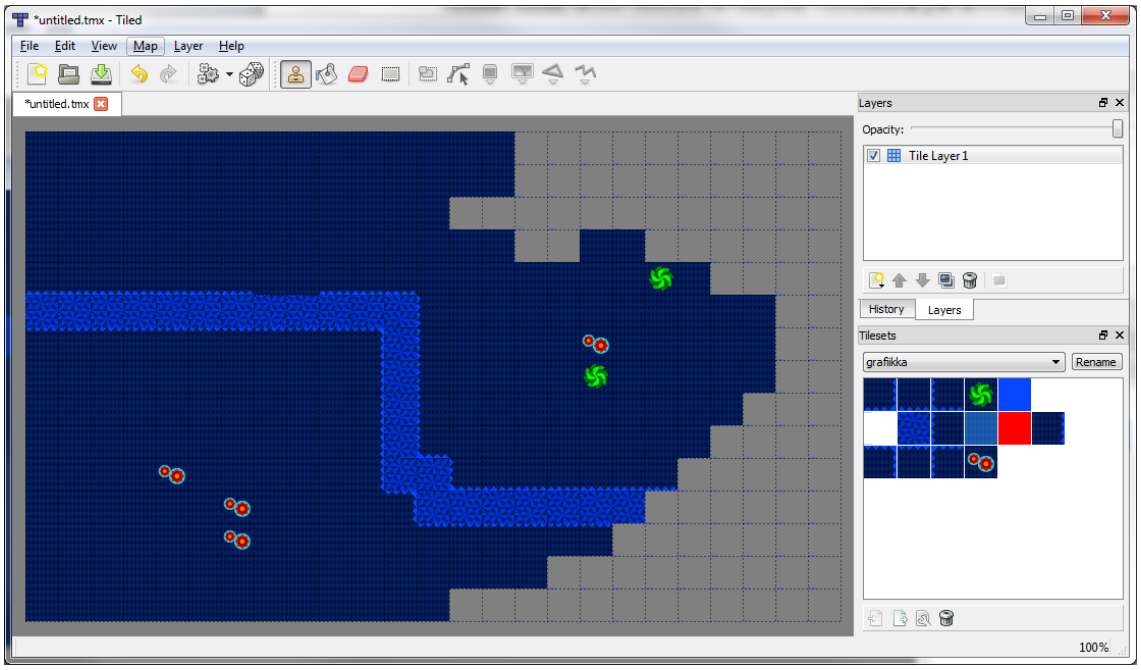
nen ympärille on rakentunut melko suuri yhteisö, jonka avulla voi etsiä apua ongelmiin. Erilaisia ohjeita ja esimerkkejä on myös saatavilla runsaasti. Tämä onkin tarpeellista sillä yksi suuri haittapuoli on AndEngine todella puutteellinen dokumentointi. (Andengine tutorials 2014.)

AndEngine otetaan käyttöön Eclipse-kehitysympäristöön Git-lisäosan avulla. Eclipsen uusimmissa versioissa se on valmiina, mutta se voidaan myös asentaa erikseen, jos sitä ei löydy valmiina. Git-lisäosa mahdollistaa ohjelmakirjastojen lataamisen suoraan tiedostovarastoista. Kun ohjelmakirjastot on ladattu, niitä voidaan käyttää omassa projektissa. (Ricardo 2013.)

5.5 Tiled Map Editor -kenttätyökalu

Tiled Map Editor on ohjelma ruutupohjaisten pelikenttien tekemistä varten. Sillä voidaan luoda halutun kokoisia kenttiä, jotka muodostuvat pienemmistä ruuduista. Yksittäisen ruudun koko voidaan määrittää itse. Ohjelmalla voidaan myös rakentaa kenttiä, jotka eivät koostu ruuduista. Tiled Map Editorilla on helppo päästä alkuun ja sen käyttämä tmx-tallennusmuoto on yhteensopiva AndEnginen kanssa. Kentän ruuduille voidaan asettaa erilaisia ominaisuuksia kuten esimerkiksi törmäyksen havainnointi. (Bruner 2012.)

Uuden kartan tekeminen aloitetaan valitsemalla orientaatio, eli halutaanko kartan olevan vaakatasossa vai kulmittain, kartan koko eli kuinka monta ruutua vaakaan ja pystyyn, sekä yksittäisen ruudun koko pikseleinä ilmoitettuna. Kun nämä asetukset on määritetty, ohjelma näyttää kenttäpohjan, joka perustuu määritettyihin asetuksiin. Tämän jälkeen tarvitaan vielä ruutujoukko (eng. tileset), josta valitaan ruudut karttaan. Tiled Map Editor osaa luoda ruutujoukon kuvasta. Luotavalle ruutujoukolle määritetään yksittäisen ruudun koko, jonka jälkeen annetaan polku kuvaan, josta ruutujoukko halutaan luoda. Kun ruutujoukko on valmis, voidaan kentän tekeminen aloittaa. Ruutujoukosta valitaan haluttuja ruutuja, jotka voidaan lisätä kenttäpohjaan (Kuva 1). Ruutuja voidaan lisätä myös moneen tasoon. (Bruner 2012.)



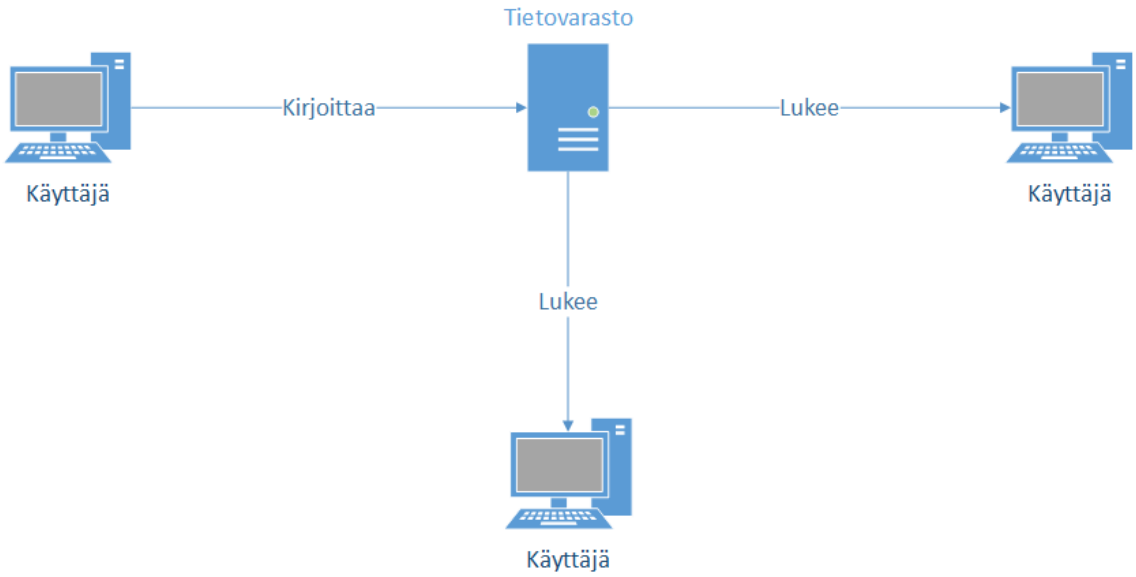
Kuva 1. Kuvankaappaus Tiled Map Editorista kentän luonnin yhteydessä.

6 Versionhallinta osana ohjelmistokehitystä

Versionhallinnan avulla pystytään seuraamaan tiedostoihin tai hakemistoihin tehtyjä muutoksia. Versionhallintajärjestelmän avulla käyttäjä tai käyttäjät voivat tarkastella muutoksia ja palata aiempiin versioihin tiedostoista, sillä järjestelmä tallentaa jokaisen version erikseen. Järjestelmän ydin on tietovarasto (eng. repository), joka sisältää versionhallinnan piiriin kuuluvat tiedostot. Tietovarasto on keskitetty säilytyspaikka tiedostoille. Tiedostot on tallennettu tietovarastoon yleensä tiedostopuun muodossa tiedostoina ja kansioina. Varastoon yhdistävät käyttäjät voivat tarkastella mitä tahansa versiota tiedostoista ja muokata niitä. (Collins-Sussman, Fitzpatrick & Pilato 2011, 1.)

6.1 Versionhallintajärjestelmä

Tietovarasto sisältää järjestelmän piiriin kuuluvat tiedostot, joita käyttäjät lukevat ja muokkaavat. Kun käyttäjä on muokannut tiedostoja, hän kirjoittaa muutokset varastoon muiden käyttäjän saataville. Lukemalla tiedostoja käyttäjä saa muiden tekemät muutokset itselleen (kuvio 1). (Collins-Sussman ym. 2011, 1.)



Kuvio 1. Kuvaus tietovaraston toiminnasta (Collins-Sussman ym. 2011).

Käyttäjät eivät koskaan muokkaa suoraan varastossa olevia tiedostoja, vaan paikallisia kopioita. Paikallinen kopio sisältää tietyn version varastossa olevista tiedostoista. Tätä paikallista kopiota käyttäjä voi muokata vapaasti, ja kun muokkaus on valmis, muokatut tiedostot voi lähettää takaisin tietovarastoon. (Collins-Sussman ym. 2011, 2.)

Yksi versionhallintajärjestelmien ratkaistavaksi jäävä ongelma on, kuinka estää käyttäjiä ylikirjoittamasta toisten tekemiä muutoksia. Jotkut järjestelmät ratkaisevat ongelman siten, että samaa tiedostoa voi muokata vain yksi käyttäjä kerrallaan. Tämä tapahtuu lukitsemalla tiedostot, joita halutaan muokata. Jos toinen käyttäjä yrittää muokata samaa tiedostoa, järjestelmä ilmoittaa asiasta eikä salli tiedoston muokkaamista. Tiedostoa voi lukea, mutta muokkaaminen onnis-

tuu vasta, kun tiedoston lukinnut käyttäjä lopettaa muokkauksen ja avaa lukituksen. (Collins-Sussman ym. 2011, 3.)

Tiedostojen lukitseminen toimii päällekkäisyyksien ehkäisyssä, mutta siinä on muutamia haittapuolia. Tiedoston lukitsija saattaa unohtaa avata lukituksen, jolloin kukaan muu ei pääse muokkaamaan sitä. Tästä syntyy turhaa odottelua ja voi kestää pitkään, ennen kuin tiedoston lukitsija saadaan avaamaan tiedosto. Toinen iso haitta on tiedostojen riippuvuudet toisistaan. Jos eri käyttäjät muokkaavat samanaikaisesti eri tiedostoja, jotka ovat riippuvaisia toisistaan, saattavat tiedostot muuttua siten, että ne eivät ole enää yhteensopivia. Käyttäjien pitäisi olla aina tietoisia toistensa tekemisistä, jotta varmistuttaisiin siitä että tiedostot ovat muokkausten jälkeenkin yhteensopivia. (Collins-Sussman ym. 2011, 4.)

Lukitusmenetelmä ei ole ainut vaihtoehto, ja monet versionhallintajärjestelmät käyttävätkin ”kopioi-muokkaa-yhdistä”-mallia (eng. copy-modify-merge) ylikirjoitusten ehkäisemiseksi. Tässä mallissa käyttäjät voivat muokata paikallisia kopioita tiedostoista. Samojen tiedostojen yhtäaikainen muokkaus on mahdollista. Muokkausten jälkeen käyttäjien paikalliset kopiot yhdistetään toisiinsa. Versionhallintajärjestelmät hoitavat tiedostojen yhdistämisen yleensä automaattisesti. Jos käyttäjät ovat muokanneet täysin samaa kohtaa jossakin tiedostossa, versionhallintajärjestelmä ilmoittaa ristiriidasta, joka täytyy selvittää manuaalisesti. Ristiriitojen selvittämiseen tarvitaan aina ihmistä, sillä sovellus ei osaa käsitellä niitä. (Collins-Sussman ym. 2011, 4–6.)

Versionhallinnalla pääsee pitkälle, mutta sekään ei pysty korvaamaan kommunikaatiota käyttäjien välillä. Mitä huonommin kommunikointi toimii, sitä enemmän syntyy ristiriitoja, jotka pitää ratkaista manuaalisesti. Lukitusmenetelmällä pystytään ehkäisemään ristiriitojen synty, mutta niiden selvittämiseen menee yleensä huomattavasti vähemmän aikaa kuin mitä kuluu lukitusmenetelmästä johtuvaan odottamiseen. Tästä syystä hyvä kommunikaatio on tärkeää. (Collins-Sussman ym. 2011, 6.)

Tiedostojen lukituksella on suuria haittapuolia, mutta sillä on myös omat vahvuutensa. Kopioi-muokkaa-yhdistä-menetelmä toimii yleensä hyvin, kun tiedostot sisältävät tekstiä. Esimerkiksi ohjelmakooditiedostojen kanssa tämä versionhallintamuoto toimii ilman suuria ongelmia. Menetelmä ei kuitenkaan ole usein mahdollista esimerkiksi tiedostojen kanssa, jotka sisältävät grafiikkaa tai ääntä. Näiden tiedostojen kanssa lukitusmenetelmä on lähes pakollinen. (Collins-Sussman ym. 2011, 6–7.)

6.2 Mercurial-versiohallinnan käyttö

Mercurial on avoimeen lähdekoodiin perustuva versionhallintajärjestelmä. Sen kehityksessä on otettu huomioon alustariippumattomuus, minkä johdosta se on saatavilla kaikille suosituimmille käyttöjärjestelmille. Tuettuja käyttöjärjestelmiä ovat mm. Windows, MacOS ja GNU/Linux. Järjestelmää voidaan käyttää komentorivikomennoilla, mutta esimerkiksi Windowsille on saatavilla myös graafinen käyttöliittymäsovellus, josta lisää myöhemmin. (Mercurial 2014b.)

Seuraavaksi käydään läpi perusteet Mercurial-versionhallinnan käytöstä. Järjestelmää voidaan käyttää komentoriviltä tai graafisen käyttöliittymän omaavilla työpöytäsovelluksilla. Seuraavissa esimerkeissä näytetään, kuinka järjestelmän käyttö tapahtuu komentoriviltä käsin. Ennen käyttöä Mercurial täytyy asentaa. Mercurialin sivuilta on saatavilla asennustiedosto, jonka avulla asentaminen tapahtuu ohjattua toimintoa käyttäen. Mercurial tulee myös TortoiseHg-työkalupaketin mukana.

Kun Mercurial-versionhallinnan käyttö aloitetaan, on syytä varmistua siitä, että asennus on onnistunut. Asennuksen onnistumisen voi tarkistaa vaikkapa tulostamalla Mercurialin versiotiedot. Versiotietojen tulostaminen tapahtuu komennolla ”hg version”. Mikäli komentoriville tulostuu versiotietoja, versionhallintajärjestelmä on asennettu onnistuneesti ja sen käyttö voidaan aloittaa. (O'Sullivan 2009.)

Mercurial tarjoaa apua komentoihin ja niiden käyttöön sisäänrakennetun tukijärjestelmän kautta. Kaikki järjestelmän komennot saadaan näkyville kirjoittamalla ”hg help”. Mikäli jostain komennosta halutaan tarkempaa tietoa, voidaan kirjoittaa ”hg help <komento>”. (O'Sullivan 2009.)

Ensimmäinen asia, joka tehdään kun versionhallinnan käyttö aloitetaan, on olemassa olevan tietovaraston kloonaus tai uuden luominen. Kloonaamalla saadaan tietovarastosta paikallinen kopio. Jos olemassa olevaa projektia tai varastoa ei ole, voidaan sellainen luoda. Valmiin tietovaraston kloonaaminen tapahtuu komennolla ”hg clone <lähdehakemisto> <kohdehakemisto>”. Kloonaaminen voidaan suorittaa paikallisesti tai verkon yli. Uuden tietovaraston luominen tapahtuu navigoimalla haluttuun tyhjään kansioon ja kirjoittamalla ”hg init”. Kun tietovarasto on luotu tai kloonattu johonkin kansioon, sinne ilmestyy kansio nimeltä .hg. Tämä kansio sisältää kopion varsinaisesta varastosta ja kaikki muut tiedostot sijaitsevat niin sanotussa työhakemistossa (eng. working directory). Tietovarasto sisältää varsinaisen versiohistorian, kun taas työhakemistossa on tietty vedos projektin tiedostoista. Kaikki muutokset, joita käyttäjä tekee, tehdään työhakemistoon. (O'Sullivan 2009.)

Komento ”hg log” näyttää versiohistorian. Versiohistoria koostuu tiedostojen eri versioista. Näitä eri versioiden kokonaisuuksia kutsutaan muutosjoukoiksi (eng. changeset) tai revisioiksi (eng. revision). Jokaisella muutosjoukolla on oma tunnistenumeronsa, jonka perusteella ne tunnistetaan. Käyttäjä voi tarkastella tiedostoja ja vertailla niitä eri muutosjoukkojen välillä. (O'Sullivan 2009.)

Kun käyttäjä muokkaa tiedostoja paikallisen kopion työhakemistossa, versionhallinta tunnistaa nämä muutokset. Enemmän tietoja muuttuneista tiedostoista saadaan komennolla ”hg status”. Tämä tulostaa kaikkien muuttuneiden tiedostojen nimet. Mikäli jonkun tiedoston muutoksia halutaan tarkastella tarkemmin, voidaan kirjoittaa ”hg diff”. Näin saadaan yksityiskohtainen tieto siitä, mitä tiedostoissa on muuttunut ja missä kohtaa tiedostoa. Kun käyttäjä on valmis tallentamaan tehdyt muutoksensa, niistä voidaan luoda uusi muutosjoukko. Uusi muutosjoukko luodaan komennolla ”hg commit”. Muutosjoukon luomista kutsutaan usein nimellä kommitointi (eng. commit) tai kommittaaminen. Kommitointi

luo uuden muutosjoukon paikalliseen kopioon. Jokaisen käyttäjän tulee luoda käyttäjätunnus, jolla kommitointi tehdään. Näin muut pystyvät näkemään, kuka kommitin on tehnyt. (O'Sullivan 2009.)

Kun käyttäjä kommittaa, hänen täytyy syöttää kommit-viesti. Kommit-viestissä kuvataan yleensä, mitä muutoksia kyseinen muutosjoukko sisältää. Näin käyttäjät näkevät eri muutosjoukkoja tarkastellessaan (”hg log” komento) mitä ne sisältävät. Kun kommit on tehty, sitä voidaan tarkastella komennolla ”hg tip”. Se näyttää tiedot aina tietovaraston uusimmasta muutosjoukosta. Uusinta muutosjoukkoa kutsutaankin nimellä tip eli ”kärki”. (O'Sullivan 2009.)

Kun uusi muutosjoukko on tehty, se voidaan siirtää paikallisesta kopioista varsinaiseen tietovarastoon muiden saataville. Uusia muutosjoukkoja, joita ei ole vielä lähetetty voi tarkastella komennolla ”hg outgoing”. Komento näyttää, mitä muutoksia tietovarastoon lähetetään. Varsinaisen siirtämisen suorittaa komento ”hg push”. (O'Sullivan 2009.)

Muiden tekemät muutokset saadaan tietovarastosta paikalliseen kopioon komennolla ”hg pull”. Yleensä kannattaa kuitenkin katsoa mitä muutoksia ollaan lataamassa ennen varsinaista latausta. Tämä tapahtuu komennolla ”hg incoming”. Jos muutoksissa ei huomata ongelmia, voidaan ne ladata paikalliseen kopioon. Uusien muutosjoukkojen lataaminen paikalliseen kopioon ei päivitä tiedostoja työhakemistoon. Työhakemiston pystyy päivittämään siihen tilaan, missä se oli minkä tahansa muutosjoukon kohdalla. Päivittäminen tapahtuu komennolla ”hg update <muutosjoukon tunnistenumero>”. Tämä päivittää työhakemiston tiedostot siihen tilaan, jossa ne olivat kyseisessä muutosjoukossa. Muutosjoukon tunnistenumeron voi jättää pois komennosta, jolloin päivitetään automaattisesti uusimpaan muutosjoukkoon. (O'Sullivan 2009.)

Usein tulee eteen tilanteita, joissa eri käyttäjät ovat muokanneet projektin tiedostoja samanaikaisesti. Jos ”hg pull” -komento lataa paikalliseen kopioon muutosjoukon, jossa ei ole paikallisessa kopiossa käyttäjän tekemiä muutoksia, käyttäjän muutokset täytyy yhdistää ladattuihin muutoksiin, jotta työhakemisto saadaan päivitettyä ajan tasalle. Yhdistäminen tapahtuu komennolla ”hg mer-

ge”. Näin yhdistetään automaattisesti viimeisin paikallinen muutosjoukko ja viimeisin ladattu muutosjoukko. Tämän jälkeen tehdään kommit, joka tekee muutosjoukon yhdistetyistä muutosjoukoista. Nyt käyttäjällä on viimeisimmät muutokset yhdistettynä muutosjoukkoon, joka voidaan lähettää varsinaiseen tietovarastoon. (O'Sullivan 2009.)

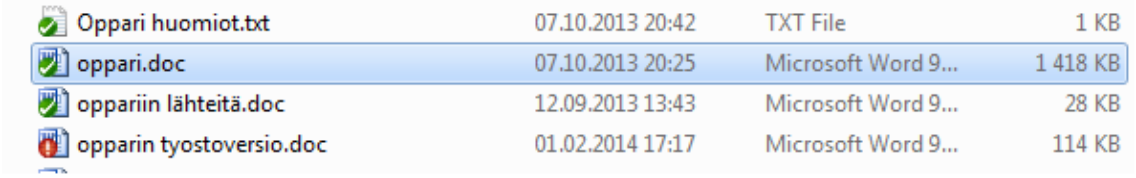
6.3 Bitbucket-web-säilytystila

Bitbucket on pilvipalvelu, joka mahdollistaa ohjelmakoodin ja sekä Mercurialettä Git-tietovarastojen säilyttämisen ja hallitsemisen pilvessä. Se on ilmainen ja sallii rajattoman määrän tietovarastoja. Palvelun ilmaisversiossa jokaisen tietovaraston voi jakaa viidelle henkilölle. Palvelua voi käyttää suoraan selaimessa ja se mahdollistaa tietovarastojen käsittelyn graafisen käyttöliittymän avulla. Kommit-historiaa pystyy selaamaan ja tarvittaessa vertailemaan tiedostoja ja niihin tehtyjä muutoksia. Lähdetiedostojen lataaminen on myös mahdollista. (Atlassian Bitbucket 2014.)

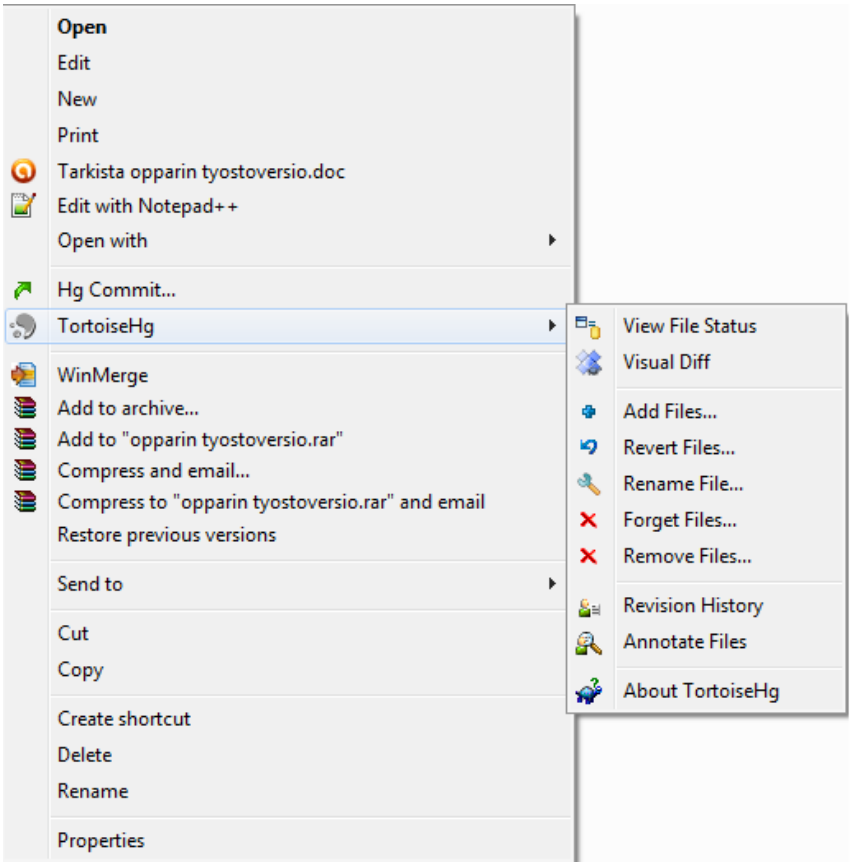
Pilvipalvelun etuina ovat riippumattomuus ja saatavuus. Kun varsinainen tietovarasto ei ole kenenkään omalla koneella, ei esimerkiksi laiteviasta tai muusta vastaavasta ole vaaraa lähdetiedostoille. Kuka tahansa tietovarastoon pääsyn omaava henkilö pystyy tarkastelemaan tiedostoja missä tahansa, kunhan nettiyhteys löytyy.

6.4 TortoiseHg graafinen työkalupaketti

TortoiseHg on graafinen työkalupaketti Mercurial-versionhallintajärjestelmälle. Erityisesti Windows-käyttöjärjestelmässä se sisältää kaiken tarvittavan Mercurialin käytön aloittamiseen. Windowsilla työkalupaketti sisältää laajennoksen, joka lisää tiedostoselaimeen ikoneita ja valikoita Mercurialin käyttämisen helpottamiseksi (kuvat 2 ja 3). (Borho 2014.)



Kuva 2. Kuvankaappaus TortoiseHg-laajennoksen lisäämistä ikoneista Windowsin tiedostoselaimeen.



Kuva 3. Kuvankaappaus TortoiseHg-laajennoksen lisäämistä valikoista Windowsin tiedostoselaimeen.

TortoiseHg:n Windows-versiossa on myös Workbench-niminen sovellus, jonka avulla pystytään käyttämään graafisen käyttöliittymän kautta edellä mainittuja Mercurialin toimintoja. Työkalupaketti sisältää myös varsinaisen Mercurial-versionhallintajärjestelmän, joten TortoiseHg-työkalupaketin asentamisen jälkeen Mercurial-järjestelmä on valmiina käytettäväksi. (Borho 2014.)

7 Case: Tower defence

Kun päätimme toteuttaa opinnäytetyön käytännön osuuden yhdessä Heikki Ylösen kanssa, emme vielä tienneet mitä tekisimme. Ajatuksena alusta lähtien oli jonkun pelin prototyypin tai rungon tekeminen. Lyhyehkön pohdinnan jälkeen päätimme, että tekisimme pelin mobiilialustalle ja tarkemmin sanottuna nimenomaan Androidille. Olimme itsekin pelanneet paljon älypuhelimilla ja tableteilla, joten mielestämme aihealue vaikutti mielenkiintoiselta.

Kummallakaan meistä ei ollut aikaisempaa kokemusta pelien tekemisestä mobiililaitteille. Minulla oli koulun puolesta käytynä Android-ohjelmoinnin peruskurssi, mikä oli eräs syy juuri Androidin valikoitumiseen alustaksi. Toinen syy Androidin valinnalle oli se, että me molemmat omistimme Android-laitteen. Heikillä oli älypuhelin ja minulla tablet-tietokone. Laitteiden erilaisuus lisäsi mielenkiintoa toteutusta ajatellen, sillä emme tienneet, miten eri laitteet ja Android-versiot tulisi ottaa huomioon. Myös se, että meillä molemmilla oli kokemusta Java-ohjelmoinnista, vaikutti alustan valintaan. Ohjelmointi Androidille tapahtuu pääasiassa Javalla.

Mietimme varsinaisen pelin aihetta pari päivää. Tower defencen lisäksi ideoimme sivulta kuvattuja ja vasemmalta oikealle eteneviä pelejä kuten maastopyöräpeli. Totesimme kuitenkin, että liikkuvan kentän tekeminen olisi ehkä hieman liian haastava ensimmäiseen peliprojektiimme. Lopulta päädyimme ylhäältä päin kuvattuun tower defenceen, sillä tarkemman tutkiskelun jälkeen se vaikutti yksinkertaisimmalta toteuttaa. Emme halunneetkaan keksiä mitään uutta ja mullistavaa, vaan valitsimme jo olemassa olevan peli-idean, jonka pystyisimme toteuttamaan ilman, että joutuisimme suunnittelemaan koko pelin alusta lähtien itse.

7.1 Mikä on tower defence?

Tower defence tarkoittaa vapaasti suomennettuna tornipuolustusta. Tower defence -pelien pohjimmainen idea on estää etenevien vihollisjoukkojen pääsy paikasta A, paikkaan B. Lähes kaikki tower defence -pelit noudattavat samaa peruskaavaa, mutta pieniä eroja voi löytyä[1]. Vihollisjoukkoja tulee yleensä asteittain vaikeutuvissa sykleissä ja niiden eteneminen pyritään estämään erilaisia puolustustorneja rakentamalla (kuva 4). Vaikeusaste kasvaa joko vihollisten määrän lisääntyessä tai vihollisten tuhoamiseen tarvittavien osumien määrän kasvaessa.

Kuva 4. Kuvankaappaus tower defence -pelin prototyypistä, jonka teimme opinnäytetyön käytännön osuudessa. Laitteena Asus Nexus 7 tablet-tietokone.

Erilaisia torneja voi olla useita erilaisia ja niitä voi yleensä päivittää tehokkaammiksi. Jotkut tornit vaikuttavat vain yhteen viholliseen kerrallaan ja jotkut saattavat vaikuttaa useisiin. Myös erilaisia vihollisia on yleensä useita. Jotkut viholliset liikkuvat nopeammin kuin toiset ja jotkut voivat esimerkiksi lentää. Kaikki tornityypit eivät välttämättä pysty vaikuttamaan kaikkiin vihollistyyppeihin ja se on

[1] 10 esimerkkiä tower defence -peleistä: http://armorgames.com/news/top-10-tower-defense-games

otettava huomioon torneja rakentaessa. Uusien tornien rakentamiseen ja jo olemassa olevien päivittämiseen tarvitaan rahaa. Rahaa voi saada vihollisten tuhoamisesta, tason läpäisemisestä tai sitä voi myös kertyä pikkuhiljaa itsestään.

7.2 Ohjelmien ja toteutustapojen valinta

Tässä luvussa kerron, kuinka valitsimme projektissa käytetyt ohjelmat ja toteutustavat. Viestimisessä käytimme entuudestaan hyväksi havaittuja ja tuttuja ohjelmia. Käyttöön valikoitui myös ennestään tuntemattomia ohjelmia ja alustoja, koska aikaisempaa kokemusta pelien tekemisestä mobiilialustoille ei ollut. Yksi tällainen tuntematon tekijä oli pelin varsinainen runko eli AndEngine-pelimoottori.

7.2.1 Kehitysmenetelmä ja viestintäohjelmat

Kun suunnittelimme projektin toteutusta, mietimme, miten varsinainen käytännön työskentely toteutettaisiin. Koska kummallakaan meistä ei ollut varsinaisesti kokemusta pelikehityksestä tai sovelluskehityksestä ylipäätänsä mobiililaitteille, päätimme, että teemme suurimman osan työstä yhdessä. Ajattelimme, että yhdessä voimme nopeammin omaksua uusia asioita. Toinen meistä voisi hoitaa varsinaisen ohjelmoinnin ja toinen katsoisi ja kommentoisi samalla, ja etsisi ratkaisuja mahdollisiin ongelmiin. Suunnittelun edetessä pidemmälle tajusimme että ajattelemamme kehitysmenetelmä muistuttaa hyvin paljon, ellei lähes täysin, pariohjelmointia. Pariohjelmointi valikoitui siis melkein vahingossa kehitysmenetelmäksi.

Koska työskentelimme eri paikkakunnilta käsin, mutta kuitenkin yhdessä samanaikaisesti, tarvitsimme tehokkaan tavan reaaliaikaiseen yhteydenpitoon. Tarvitsimme varsinkin pariohjelmointi-kehitysmenetelmän takia jonkun sovelluksen, jonka avulla saamme jaettua työpöytänäkymän toinen toisillemme. Työpöy-

tänäkymän jakaminen on yksi Adobe Connectin keskeisimmistä ominaisuuksista ja koska meillä molemmilla oli sovelluksesta jo kokemusta koulun puolesta, päätimme käyttää ensisijaisesti sitä pariohjelmoinnin mahdollistamiseksi. Meillä oli koulun puolesta valmiit Adobe Connect -huoneet, sekä käyttäjätunnukset jo olemassa, joten sovelluksen käyttöönottoon ei kuluisi turhaa aikaa.

Vaikka tiedostojen ja viestien lähettäminen onnistuu Adobe Connectissa, päätimme käyttää viestinnässä ja yhteydenpidossa myös Skypeä. Adobe Connectin käyttäminen vaatii aina sen käynnistämisen verkkoselaimen kautta, ja huoneen avaamisen. Tämän jälkeen myös toisen käyttäjän täytyy liittyä huoneeseen keskustelun käymiseksi. Tästä syystä päätimme käyttää Skypeä silloin, kun emme tehneet varsinaista pariohjelmointia, tai muuta toteutusta. Meillä kummallakin on Skype päällä lähes koko ajan ja koska lähetetyt viestit jäävät muistiin, ei haittaa vaikka toinen ei olisikaan paikalla juuri silloin kun viestin lähettää. Skype oli luonnollinen valinta osittain samoista syistä, kuin Adobe Connect. Se oli molemmilla jo asennettuna, ja olimme käyttäneet sitä keskenämme viestimiseen jo aikaisemminkin. Ohjelman ominaisuudet olivat meille entuudestaan tuttuja ja tiesimme, että se soveltuu käytettäväksi myös opinnäytetyö projektissa. Viestintäohjelmien tarkempi vertailu vaadittujen ominaisuuksien pohjalta löytyy seuraavassa taulukossa (taulukko 2).

	Adobe Connect	Skype	Mumble	Sähköposti
Ääniviestintä	X	X	X	
Ruudunjako	X	X		
Etähallinta	X			
Tekstipohjaiset viesti	X	X	X	X
Tiedostojen lähetys	X	X		X
Aiempi kokemus	X	X	X	X

Taulukko 2. Viestintäohjelmien vertailu vaadittujen ominaisuuksien pohjalta.

7.2.2 Kehitysympäristön ja pelimoottorin valinta

Koska Android-kehityksen mahdollistava Android Developer Tools -työkalupaketti on lisäosa Eclipse-kehitysympäristöön, valikoitui Eclipse oikeastaan automaattisesti projektin kehitysympäristöksi. Sovellusten ja pelien kehitys Androidille on mahdollista ilman varsinaista kehitysympäristöä, mutta päättelimme, että Eclipsen kanssa se on helpompaa. Löysimme myös paljon oppaita, jotka neuvovat, kuinka Android-kehittäminen tapahtuu käyttäen Eclipseä[2]. Eräs merkittävä tekijä, jonka vuoksi valitsimme Eclipsen, oli se, että se näytti olevan hyvinkin toimiva kehitysympäristö käytettäväksi AndEngine-pelimoottorin kanssa. Koimme toimivimmaksi ratkaisuksi Eclipsen sen takia, että hyvin suuressa osassa, ellei jopa kaikissa löytämissämme AndEngine-oppaissa käytettiin juuri Eclipseä. Koska kummallakaan meistä ei ollut aikaisempaa kokemusta AndEnginestä, meistä tuntui järkevältä valita yleisimmin käytössä oleva kehitysympäristö myös omaan projektiimme.

Eclipsen valintaa puolsi myös se, että sekä minulla että Heikillä oli aiempaa kokemusta siitä. Itse olen käyttänyt sitä pitkään yhtenä kehitysympäristönä töissä. Olen käyttänyt Eclipseä lähes päivittäin viimeisen kahden vuoden aikana, joten koin, että minulla on melko vankka osaaminen liittyen siihen. Eclipse oli käytössä myös Android-ohjelmoinnin peruskurssilla, jonka olin käynyt koulussa aiemmin. Tästä syystä ajattelimme, että siitä kokemuksesta olisi varmasti paljon hyötyä tässä peliprojektissa.

AndEnginen valikoituminen pelimoottoriksi tulikin jo edellisissä luvuissa esille. Ensimmäisenä pelimoottoria valitessa meille tuli mieleen Unity. Unity on suosittu pelimoottori, jolla voidaan tehdä 2D- ja 3D-pelejä. Varsinaista kokemusta meillä ei siitä ollut, mutta se oli tullut usein vastaan koulussa. Tiesimme myös muutamia pelejä, jotka oli toteutettu käyttäen sitä. Unityn ongelmaksi muodostui kuitenkin se, että se oli maksullinen. Minulla oli koulun puolesta Android- sekä iOS -lisenssi olemassa, mutta Heikillä sitä ei ollut. Unitystä on saatavilla 30 päivän kokeiluversion, mutta tiesimme, että se olisi liian lyhyt aika saada projekti

[2] Kattava ohje Android-kehitykseen: http://www.vogella.com/tutorials/Android/article.html

valmiiksi. Halusimme myös mahdollisesti jatkaa pelin kehitystä opinnäytetyön ulkopuolella, joten ilmainen pelimoottori sopisi tähän projektiin parhaiten. Pelimoottorin valinnasta tarkempi taulukko alla (taulukko 3).

	AndEngine	Unity
Mahdollista toteuttaa halutut toiminnallisuudet	X	X
Ilmainen	X	
Aiempi kokemus		
Hyvä dokumentaatio		X
Saatavilla hyviä ohjeita	X	X

Taulukko 3. Unityn ja AndEnginen vertailu.

AndEnginestä kuulimme ensimmäisen kerran jo kauan ennen tämän projektin alkua. Eräs tuttumme oli käyttänyt sitä omassa opinnäytetyöprojektissaan. Hänen projektinsa tuloksena syntynyt peli oli hyvin samantyylinen kuin se, minkä me lopulta päätimme toteuttaa. Kun olimme rajanneet aihealueen suurin piirtein ylhäältä päin kuvattuun tower defence -peliin, kyselimme hieman muilta samaa alaa opiskelevilta tutuiltamme, oliko heillä kokemusta pelimoottoreista, ja voisivatko he suositella jotain. Tässä yhteydessä AndEngine tuli esille.

Ryhdyimme tutkimaan AndEngineä tarkemmin, ja se tuntuikin soveltuvan käyttötarkoitukseemme hyvin. AndEngineen tutustuessamme törmäsimme Tiled Map Editor nimiseen ohjelmaan, jolla pystyi tekemään tiilipohjaisia pelikenttiä, joita AndEngine tukee. Koska huomasimme, että Tiled Map Editorilla, ja AndEnginellä pystyisimme toteuttamaan kaikki ominaisuudet, joita olimme miettineet, päädyimme melko lyhyen pohtimisen jälkeen käyttämään niitä. Eräs asia, mistä olimme hieman huolissamme, oli AndEnginen todella puutteellinen dokumentointi. Pysyimme kuitenkin valinnassamme, sillä AndEnginelle tuntui löyty-

vän hyviä oppaita melko paljon. AndEnginellä on myös hyvin aktiivinen yhteisö ja sen foorumeilta löytyi paljon vinkkejä ja apua[3].

Edellä mainittu Tiled Map Editoriin törmäsimme, kun tutustuimme AndEngine oppaisiin. Se vaikutti sopivalta ja aloimme selvittää tarkemmin sen ominaisuuksia ja mitä sillä voisi tehdä. Emme olleet pohtineet vielä miten tekisimme peliin vaaditut kentät ja Tiled Map Editor näytti tarjoavan juuri ne ominaisuudet, mitä tarvitsisimme. Sen ominaisuuksien pohjalta saimme myös ideoita itse pelin toteutukseen. Kentät toteutettaisiin yksittäisistä ruuduista, joille pystyi antamaan ominaisuuksia. Ajattelimme, että näin pystyisimme helposti määrittelemään kenttään aloitus- ja lopetusruudut. Voisimme myös määritellä missä ruuduissa vihollisjoukot kulkevat ja mihin ruutuihin torneja saa rakentaa.

7.2.3 Versionhallintajärjestelmä

Kuten monen muukin sovelluksen ja toteutustavan kohdalla, valitsimme versionhallintajärjestelmän vahvasti aikaisemman kokemuksen perusteella. Mercurial ja siihen saatavilla olevat sovellukset olivat minulle tuttuja ja käytössä lähes joka päivä. Versionhallintajärjestelmä on tärkeä osa mitä tahansa vähänkin suurempaa projektia ja emme halunneet opetella uuden järjestelmän käyttöä, sillä projektissa oli jo tarpeeksi uusia tekniikoita ja ohjelmia, joiden opettelemiseen kuluisi aikaa. Alla taulukko versionhallintajärjestelmien verailusta (taulukko 4).

	Mercurial	SVN	Git
Ohjelmakoodin lähettäminen verkkotietovarastoon	X	X	X
Entuudestaan tuttu	X		
Bitbucket yhteensopiva	X		X

Taulukko 4. Vertailu versionhallintajärjestelmien välillä.

[3] AndEnginen foorumit: http://www.andengine.org/forums/

Tarvitsimme tietovaraston säilytyspaikan verkossa, jotta pystyisimme helposti lähettämään omat muutokset toiselle ja lataamaan toisen tekemät muutokset itsellemme. Tähän tarkoitukseen loimme tietovarastolle säilytyspaikan Bitbucket pilvipalveluun. Minulla oli käyttäjätunnus valmiina kyseisessä palvelussa, ja muidenkin projektien tietovarastoja säilytyksessä siellä. Näistä syistä oli luonnollista käyttää Bitbucketia myös opinnäytetyön tietovaraston säilyttämiseen.

7.3 Tower defence -pelin toteutus ja havaintoja käytetyistä ohjelmista

Tässä luvussa käydään läpi miten valittuja ohjelmia ja työskentelytapoja käytettiin opinnäytetyöprojektissa. Pyrin myös vastaamaan tutkimuskysymyksiin kunkin ohjelman osalta. Tämän opinnäytetyön tutkimuskysymykset olivat seuraavat:

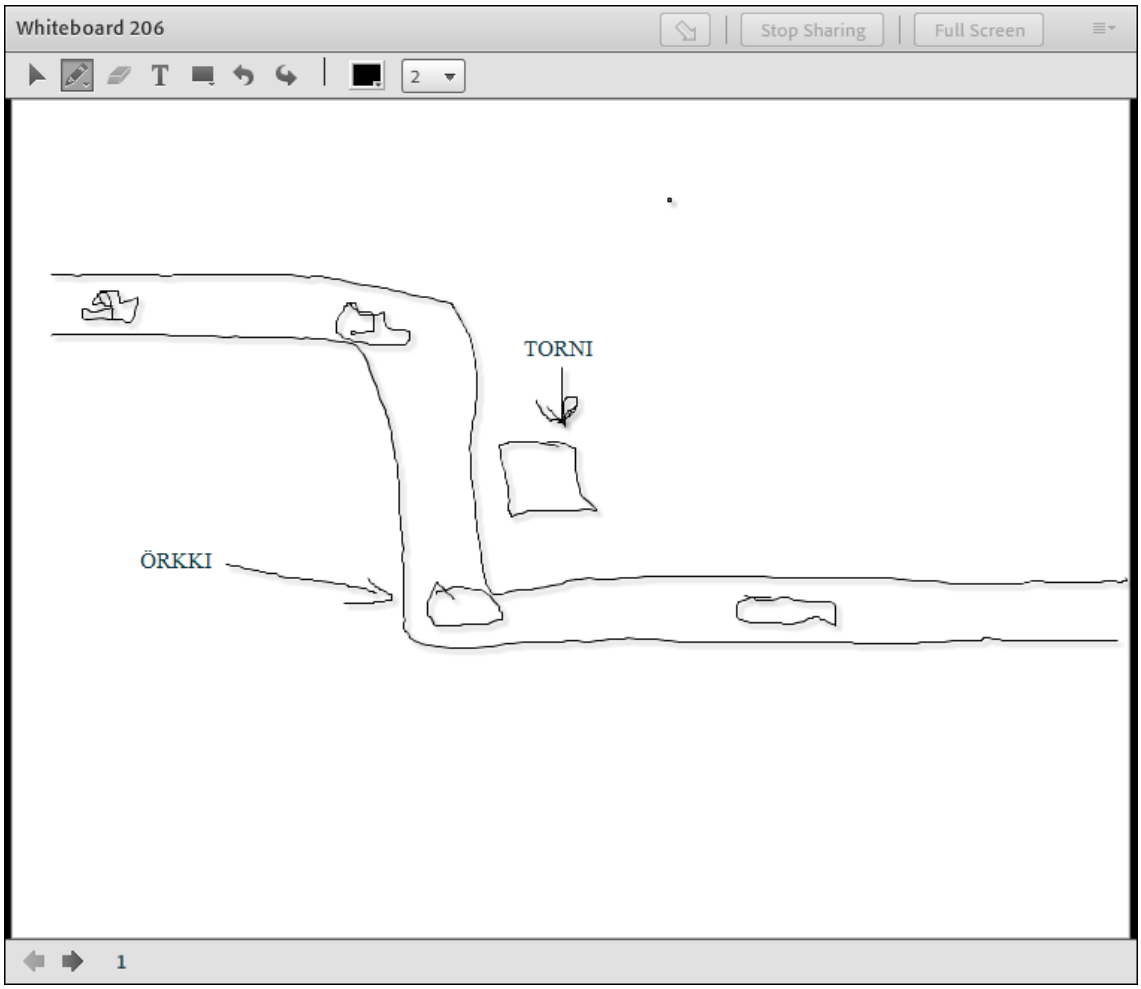
- Miten käytetyt ohjelmat soveltuvat etätyöskentelyyn?
- Mitä hyviä ja huonoja puolia käytetyillä ohjelmilla on?
- Miten käytetyt ohjelmat soveltuvat Android-pelikehitykseen?

7.3.1 Adobe Connect ja Skype viestinnässä

Eräs suurimmista syistä, miksi päädyimme käyttämään Adobe Connectia, on sen mahdollisuus antaa itse jakamansa työpöytä tai sovellus toisen hallittavaksi. Toisen käyttäjän työpöydän tai sovelluksen etähallinta osoittautui hyödylliseksi pariohjelmoinnin toteuttamisessa etänä, koska välttämättä pelkällä puheella ei omaa asiaa pystynyt selittämään tarpeeksi hyvin toiselle. Monesti oli myös nopeampaa näyttää oma idea tai korjausehdotus toiselle, kuin yrittää pelkästään selittää niitä. Saman asian olisi voinut hoitaa myös ilman etähallintaa. Ohjelmointivuorossa oleva henkilö voisi lopettaa oman jakonsa ja ohjelmointia seurannut ja kommentoinut henkilö voisi jakaa näyttönsä selittämisen ajaksi. Tämän jälkeen olisi kuitenkin taas pitänyt vaihtaa ohjelmointivuorossa olevan henkilön työpöytä jaettavaksi, joten etähallinta on paljon nopeampaa ja vaivatto-

mampaa. Haittapuolena etähallinnassa on se, että miellyttävän käytön varmistamiseksi internet-yhteyden täytyy olla melko nopea. Esimerkiksi tapauksissa, joissa toinen osapuoli oli liittynyt Adobe Connect huoneeseen langattoman verkon kautta, jaettuna olevalla työpöydällä tapahtunut toiminta näytti tökkivältä ja selvää viivettä oli havaittavissa.

Projektin suunnitteluvaiheessa Adobe Connectin whiteboard- ominaisuus oli kätevä (kuva 5). Whiteboard on jaettava alusta, johon käyttäjät voivat kirjoittaa ja piirtää samanaikaisesti (Adobe help 2013, 33). Whiteboardeihin hahmottelimme pelin ulkoasua kuten valikkoja, tasoja ja torneja sekä vihollisyksiköitä. Kirjasimme ylös myös mahdollisia ominaisuuksia ja toteutustapoja.



Kuva 5. Kuvankaappaus Adobe Connectin Whiteboard-ominaisuudesta.

Adobe Connect toimi monella tapaa loistavasti projektityöskentelyssä edellä mainittujen ominaisuuksien takia, mutta äänen avulla kommunikointi oli hieman

hankalaa. Sovellus mahdollistaa äänen lähettämisen ja vastaanottamisen käyttäen VoIP-tekniikkaa ja käyttäjien mikrofoneja (Adobe help 2013, 122). VoIP:lla tarkoitetaan puhelujen tai äänen kuljettamista internet protokollan avulla verkossa (Cisco 2014). Oman mikrofonin saa kytkettyä helposti, ja sen säätäminen on helppoa ohjatun toiminnan avulla, mutta itse käyttäminen ei soveltunut kovin hyvin työskentelyymme. Mikrofonia ei pysty vaientamaan tai avaamaan miten-kään muuten, kuin painamalla hiirellä sovelluksessa olevaa kuvaketta. Koska teimme töitä suurimmaksi osaksi kotona, mikrofoni ei voi olla päällä jatkuvasti, koska taustamelu saattaa häiritä työskentelyä. Tästä syystä mikrofoni pitäisi pystyä vaientamaan ja avaamaan kätevämmin esimerkiksi näppäimistön näppäimestä ilman, että käyttäjän täytyy navigoida sovellukseen. Sovelluksista toiseen siirtyminen häiritsee keskittymistä esimerkiksi ohjelmoidessa. Skypessä äänen lähettäminen on mahdollista asettaa näppäinpainalluksen taakse.

Toinen haittapuoli esiintyi tiedostojen jakamisessa. Projektin aikana lähetimme toisillemme lukuisia tiedostoja kuten dokumentteja, kuvia ja jopa ohjelmakoodia sisältäviä tiedostoja. Tiedostojen jako onnistuu hyvin Adobe Connectilla, mutta koska käytössämme oli koulun huoneet, emme halunneet jakaa tiedostojamme huoneessa, joihin monella muullakin henkilöllä oli pääsy. Tiedostot olisi ollut mahdollista poistaa huoneen tallennustilasta, mutta niiden jakaminen esimerkiksi Skypen kahdenkeskisessä keskustelussa tuntui helpommalta.

Adobe Connectin ehkä suurin haittapuoli on se, että se on maksullinen. Meidän tapauksessa tätä ongelmaa ei ollut, koska pystyimme käyttämään koulun tarjoamia huoneita. Adobe Connectissa on paljon hyviä ominaisuuksia, joiden takia siitä voisi hyvinkin maksaa, mutta koin, että Android-kehityksessä käyttämämme ominaisuudet löytyvät myös ilmaisista ohjelmista. Skype on hyvä esimerkki tällaisesta ohjelmasta.

Skype häviää ominaisuuksien määrässä Adobe Connectille, mutta kahden ihmisen Anroid-projektissa moni näistä ominaisuuksista on melko turhia. Tulimme toimeen yllättävän hyvin pelkästään Skypen ruudunjaolla, ääniviestimisellä ja tiedostojen lähetysmahdollisuudella ja käytimmekin sitä melko paljon Adobe Connectin vaihtoehtona. Jos esimerkiksi hoidettavana oli joku pikkuasia, kuten

nopea palaveri, emme jaksaneet aloittaa Adobe Connect- tapaamista. Skypessä soittaminen käy todella nopeasti, varsinkin kun molemmilla se on auki lähes aina silloin, kun tietokone on päällä.

Skypessä ei ole etähallintamahdollisuutta, mutta etähallintaan on olemassa monia erillisiä sovelluksia. Esimerkiksi Windows 7 käyttöjärjestelmässä sellainen on valmiiksi saatavilla[4]. Suunnitteluvaiheessa käyttämämme Adobe Connectin whiteboard-ominaisuudenkin voi korvata vaikkapa google driven piirroksilla[5], mutta Adobe Connectin hyvänä puolena on se, että kaikki tarpeellinen on samassa paketissa, eikä tarvitse availla monia eri ohjelmia. Kaikki myös tallentuu Adobe Connect huoneeseen koostetusti.

Adobe Connect tarjoaa paljon ominaisuuksia, ja se on todella toimiva vaihtoehto yhteydenpitoon etätyöskentelyssä ja varsinkin pariohjelmointiin siitä on paljon apua. Skypen hyvä puoli on sen ilmaisuus ja erillisten lisäohjelmien avulla sillä pystyy hoitamaan samat asiat, kuin Adobe Connectilla. Jos Adobe Connectin hinta ei haittaa, tai jos se on muuten saatavilla, se on varteenotettava vaihtoehto yhteydenpitoon. Ilmaisilla vaihtoehdoillakin tulee toimeen, jos jaksaa nähdä hieman enemmän vaivaa.

7.3.2 Eclipse ja Android Developer Tools kehitysympäristönä

Eclipse ja siihen saatava Andoid Developer Tools -lisäosa ovat yhdessä mielestäni kaikkein järkevin valinta Android-kehitysympäristöksi (kts. taulukko 5), varsinkin, jos aiempaa kokemusta ei ole, tai haluaa päästä helpolla. Minulla oli jo valmiiksi asennettuna Eclipse, joten latasin erikseen Android SDK:n (Android Software Development Kit), joka sisältää tarvittavat työkalut ja ohjelmakirjastot. Tämän jälkeen asensin Eclipseen Android Developer Tools -lisäosan suoraan

[4] Tietoa Windowsin etähallinnasta: http://windows.microsoft.com/fi-fi/windows/what-is-windows-remote-assistance#1TC=windows-7

[5] Tietoa Google Drive -piirroksista: https://support.google.com/drive/topic/1360903?hl=en&ref_topic=2811744

Eclipsen kautta ja asennuksen jälkeen valitsin käytettäväksi aiemmin ladatun Android SDK:n.

	Eclipse+ ADT	Android Studio
Android kehitys mahdollista	X	X
Aiempi kokemus	X	
Yhteensopia AndEnginen kanssa	X	X
Suosittu AndEnginen käytössä	X	
Ei kehitysvaiheessa	X	

Taulukko 5. Kehitysympäristöjen vertailu.

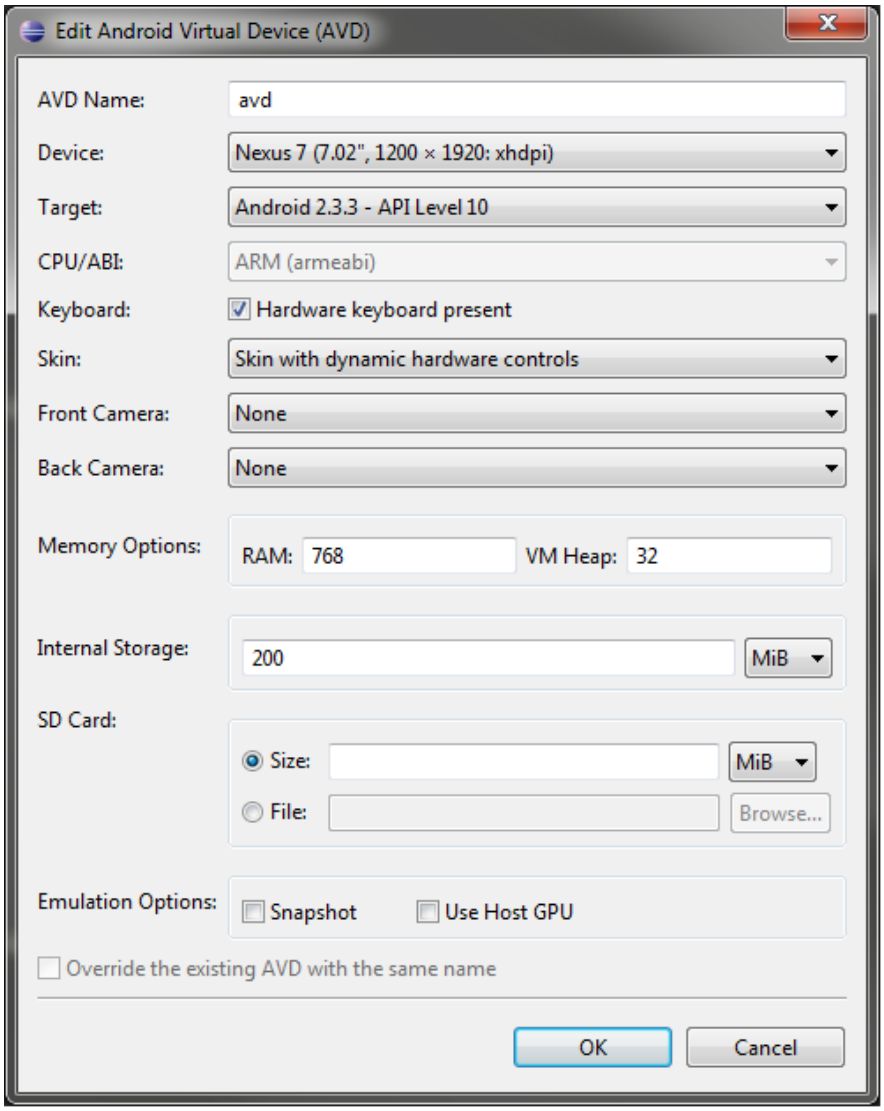
Kaikki edellä mainittu on saatavilla myös yhdessä paketissa, jonka nimi on Android ADT Bundle. Se sisältää Eclipse version, johon on valmiiksi asennettu Android Developer Tools -lisäosa, sekä Android SDK:n. Android ADT Bundle ziptiedosto ladataan koneelle ja puretaan haluttuun kansioon. Muuta ei tarvitse tehdä ja Android-kehitys voidaan aloittaa. Meidän tapauksessamme täytyi ladata vielä AndEngine-pelimoottori, jotta saisimme käyttöön tarpeelliset ohjelmakirjastot pelimme toteuttamista varten.

Eclipse itsessään sopii Android-kehitykseen varmasti yhtä hyvin, kuin mikä tahansa muukin kehitysympäristö, mutta sen yhteensopivuus ja tuki kaikelle, mitä nimenomaan Android-kehityksessä tarvitaan, on sen ehdottomasti paras puoli. Opinnäytetyön kirjoitushetkellä huomasin, että kehitteillä on Android Studio niminen ohjelma, joka ilmeisesti tulisi vaihtoehdoksi Eclipse ja Android Developer Tools -yhdistelmälle. Android Studio on vielä kehitysvaiheessa, joten ainakin kirjoitushetkellä varmin valinta kehitysympäristöksi on Eclipse.

Android Developer Tools -lisäosa tuo Eclipseen paljon käteviä ominaisuuksia Android-kehityksen helpottamiseksi. Valikkojen tekemiseksi on tarjolla graafinen työkalu, jonka avulla valikoiden luominen onnistuu helpommin, kuin niiden te-

keminen pelkästään ohjelmakoodin avulla. Tämän projektin puitteissa emme käyttäneet graafista valikkotyökalua, koska pelin prototyypin valikot olivat hyvin yksinkertaiset ja niitä oli vain kaksi. Pelin jatkokehitystä ajatellen graafinen työkalu olisi kuitenkin todella hyödyllinen, sillä suunnittelemamme ominaisuudet vaatisivat monimutkaisempien valikoiden tekemistä. Tulevaisuudessa esimerkiksi tornien ostaminen ja niiden päivittäminen tapahtuisi jonkinlaisen valikkorakenteen avulla.

Ehdottomasti paras ominaisuus, jonka Android Developer Tools tarjoaa, on testaaminen fyysisellä tai virtuaali-android -laitteella. Virtuaaliseen Android-laitteeseen voi ladata minkä tahansa Android-version, joten oman sovelluksen voi räätälöidä yhteensopivaksi mahdollisimman monen laitteen kanssa. Tämä on loistava ominaisuus, sillä sovellusten testaaminen jokaisella Android-versiolla fyysisillä laitteilla olisi todella työlästä. Harvalla yksityisellä henkilöllä on varaa ostaa fyysinen laite jokaista eri Android-versiota varten, ja yhteen laitteeseen eri versioiden asentaminen ja poistaminen olisi todella työlästä. Virtuaalilaitteen avulla kaikki tämä onnistuu helposti ja virtuaalilaitteeseen voidaan määritellä erilaisia ominaisuuksia kuten muistin määrä, prosessorin tyyppi ja näytön koko (kuva 6). Varsinkin erikokoisilla näytöillä testaaminen on hyvä tapa selvittää, miltä sovellus näyttää eri resoluutioilla.



Kuva 6. Virtuaaliselle Android-laitteelle määriteltävissä olevat ominaisuudet.

Virtuaalinen Android-laite on todella hyvä ja monipuolinen ominaisuus testaamiseen, mutta siinä on yksi suuri huono puoli, joka on sen hidas käynnistyminen. Minun tietokoneella virtuaalilaitteen käynnistymiseen kului aikaa yli minuutti. Hidas käynnistyminen alkaa häiritä nopeasti, kun säädetään jotain pientä asiaa, joka vaatii paljon muutoksia ja testaamista. Tästä syystä päädyinkin käyttämään omaa Tablet-tietokonettani testaamisessa. Virtuaalilaitteella on myös mahdotonta testata sovelluksia, joissa hyödynnetään laitteen kiihtyvyysantureita tai gyroskooppia. Gyroskoopilla pystytään havainnoimaan laitteen asentoa ja asennon muutoksia.

Jotta omaa laitetta pystyy käyttämään testaamisessa Windowsilla, täytyy asentaa usb-ajurit. Ajurien asentamisen jälkeen omasta laitteesta täytyy asettaa via-

netsintätila (eng. debug mode) päälle. Vianetsintätilan saa päälle sovelluskehittäjien asetuksista. Sovelluskehittäjien asetukset ovat piilotettu Android-versiossa 4.2 ja uudemmissa, ja niihin käsiksi pääseminen edellyttää hieman erikoisiakin toimia. Sovelluskehittäjien asetukset on mahdollista saada näkyviin menemällä laitteen asetuksiin ja painamalla laitteen koontiversionumeroa (eng. build number) seitsemän kertaa peräkkäin. Tämän jälkeen laite ilmoittaa, että kehittäjätila on otettu käyttöön. Kehittäjätilan käyttöönoton jälkeen vianetsintätila voidaan laittaa päälle ja laitetta voi käyttää testaamiseen.

Fyysinen laite kytketään tietokoneeseen usb-johdolla ja testattava sovellus käynnistyy lähes välittömästi, mikä on todella paljon mukavampaa, kuin odotella virtuaalilaitteen käynnistymistä minuutti. Omalla laitteella testaamisessa on myös se hyvä puoli, että näkee miten sovellus toimii oikeilla sormen painalluksilla ja liikkeillä. Virtuaalilaitteella on käytettävä hiirtä, mikä on tietysti hyvin erilaista, kuin laitteen käyttäminen sormilla.

Android Developer Tools yhdessä Eclipsen kanssa on ainakin tällä hetkellä mielestäni todella toimiva ratkaisu ja oikeastaan ainut järkevä vaihtoehto Anrdoid-kehittämiseen. Android ADT bundlessa on koostetusti kaikki, mitä tarvitaan sovellusten kehittämisen aloittamiseen. Eclipse sinällään ei tarjoa mitään mullistavaa muihin kehitysympäristöihin verrattuna, mutta siihen saatavat Android-työkalut tekevät siitä mielestäni parhaan vaihtoehdon. Android Developer Tools -lisäosa on ehdoton, sillä virtuaalilaitteella monipuolinen testaaminen on tärkeää, kun halutaan varmistua siitä, että oma sovellus toimii halutuilla laitteilla ja Android-versioilla. Virtuaalilaitteen hitauden vuoksi on kuitenkin suositeltavaa hoitaa jokapäiväinen testaaminen fyysisellä laitteella. Jos fyysistä laitetta ei ole olemassa, suosittelen hankkimaan sellaisen, tai kokeilemaan vaihtoehtoisia virtuaalilaitteita. Esimerkiksi Oraclen VirtualBox-sovelluksella on mahdollista ajaa Androidia[6].

[6] Ohjeet Androidin asentamiseen VirtualBoxille: http://www.howtogeek.com/164570/how-to-install-android-in-virtualbox/

7.3.3 AndEngine pelimoottorina ja Tiled Map Editor kenttätyökaluna

Pelin alustaksi valitusta AndEngine-pelimoottorin käytöstä meillä ei ollut ennakkoon oikeastaan mitään tarkempaa tietoa. Kun valitsimme projektissa käytettävää pelimoottoria, meille selvisi, että AndEnginellä ja Tiled Map Editorilla olisi teoriassa mahdollista toteuttaa haluamamme ominaisuudet. Se, miten haluttujen ominaisuuksien toteutus käytännössä tapahtuisi, oli meille täysi mysteeri.

AndEnginen asentaminen oli melko yksinkertaista ja siihen löytyi netistä paljon oppaita. Käytännössä AndEngine ladataan koneelle ja avataan projektiksi Eclipseen. Tämän jälkeen se liitetään omaan peliprojektiin kirjastona, jonka jälkeen kaikki AndEnginen toiminnallisuus on käytettävissä omassa projektissa.

AndEnginen käytössä pelotti etukäteen se, että siitä puuttui dokumentointi lähes kokonaan. Dokumentoinnin puuttuminen aiheuttikin melko paljon työtä, sillä esimerkiksi kaikkien funktioiden kohdalla ei ollut selvää, mitä parametreja niille piti antaa. Seuraavassa esimerkki vaikeaselkoisesta funktiosta, jonka käytössä dokumentoinnista olisi ollut hyötyä:

public *Path findPath(****final IPathFinderMap****<****T****>* *pPathFinderMap,* ***final int*** *pXMin,* ***final int*** *pYMin,* ***final int*** *pXMax,* ***final int*** *pYMax,* ***final T*** *pEntity,* ***final int*** *pFromX,* ***final int*** *pFromY,* ***final int*** *pToX,* ***final int*** *pToY,* ***final boolean*** *pAllowDiagonal,* ***final IAStarHeuristic****<****T****>* *pAStarHeuristic,* ***final ICostFunction****<****T****>* *pCostFunction)*

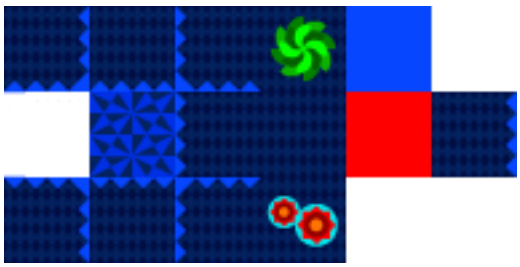
Tätä funktiota käytetään reitinetsinnässä (eng. path finding). Reitinetsinnän avulla määritellään reitti, jota pitkin vihollisjoukot etenevät aloitusruudusta lopetusruutuun. Parametrien nimistä pystyi jonkin verran päättelemään, mitä arvoja funktioihin tulisi asettaa, mutta oudompien parametrien tapauksessa se ei ollut mahdollista.

AndEnginen foorumeilta löytyy paljon ohjeita ja esimerkkiprojekteja, joiden avulla pystyimme selvittämään dokumentoinnin puuttumisesta johtuvat ongelmat. AndEnginen aktiivinen yhteisö onkin yksi sen hyvistä puolista. Apua löytyi hyvin

ja melkein kaikkiin meillä eteen tulleisiin ongelmiin oli joku muukin törmännyt jo aiemmin ja vastaukset löytyivät yleensä pienen etsimisen jälkeen. Joidenkin asioiden kuten funktioiden parametrien merkitykset olisi tosin varmasti helpompi tarkistaa kunnollisesta dokumentaatiosta, kuin foorumeilta etsimällä.

AndEnginen yksi hyvä puoli on sen avoimuus. Koska se on avointa lähdekoodia, sitä pystyy itse tarvittaessa muokkaamaan. Pelin tekemisen aikana tulikin tarve muokata AndEngineä, koska sen Sprite-luokassa, jota käytimme esimerkiksi vihollisten piirtämiseen, ei ollut funktiota kuvan vaihtamiseen. Halusimme vaihtaa vihollisen ulkomuotoa sen perusteella mihin suuntaan se liikkui.

Kenttien tekemiseen tarkoitettu Tiled Map Editor osoittautui todella toimivaksi käytössämme. Ohjelmalla pystyi luomaan kuvasta automaattisesti ruutujoukon. Määrittelimme kentälle kooksi 25 kertaa 15 ruutua ja yksittäisen ruudun kooksi 32 kertaa 32 pikseliä. Tiled Map Editor loi näillä tiedoilla uuden ruutujoukon (kuva 7), jonka jälkeen kentän tekemisen voi aloittaa. Tekeminen tapahtuu valitsemalla ruutujoukosta haluttu ruutu, jonka haluaa lisätä kenttään. Ruutuja voi laittaa useisiin tasoihin, mutta tässä projektissa pärjäsimme yhdellä tasolla. Tiled Map Editorilla tehdyt kentät tallentuvat tmx-muotoon. Tmx-kenttien käyttämiseen AndEnginessä tarvitaan AndEngineTMXTiledMapExtension-lisäosa, joka liitetään AndEngine projektiin kirjastoksi.



Kuva 7. Kenttien tekemiseen käytetty ruutujoukko.

Ennen kuin aloimme luomaan kenttiä, asetimme ruutujoukon ruuduille ominaisuuksia, jotka auttaisivat pelin toiminnassa. Jokaiselle ruudulle pystyi antamaan useita ominaisuuksia. Asetimme jokaiselle ruudulle kolme ominaisuutta, joiden

avulla saisimme kaikki prototyyppiin tulevat toiminnot tehtyä. Ominaisuudet olivat:

- isBlocked, onko ruudun päällä mahdollista kävellä vai ei.
- end, onko ruutu lopetusruutu johon vihollisjoukot yrittävät edetä.
- start, onko ruutu aloitusruutu, josta vihollisjoukot aloittavat etenemisen.

Kaikkien ominaisuuksien mahdolliset arvot olivat tosi tai epätosi (eng. true/false). Seuraavassa kuvaus pelin toiminnallisuuden kannalta tärkeimpien ruutujen ominaisuuksista:

 Polkuruutu, jolla merkattiin reitti, jota pitkin vihollisjoukot etenevät. Ominaisuudet:

- isBocked: false.
- end: false.
- start: false.

 Aloitusruutu, josta vihollisjoukot aloittavat etenemisen. Ominaisuudet:

- isBocked: false.
- end: false.
- start: true.

 Lopetusruutu, jota kohti vihollisjoukot etenevät. Ominaisuudet:

- isBocked: false.
- end: true.
- start: false.

Kaikkien muiden ruutujen omaisuuksille annettiin arvot:

- isBocked: true.
- end: false.
- start: false.

Määrittelimme ruuduille ominaisuudet, koska vihollisjoukkojen eteneminen on mahdollista toteuttaa AndEnginessä olevalla reitinetsinnällä. Jotta vihollisjoukko saatiin kulkemaan haluttua reittiä pitkin, tarvittiin aloitus ja lopetusruutujen x ja y koordinaatit, sekä tieto jokaisen ruudun kohdalla voiko sen päällä kulkea vai ei. Koordinaatit pystytään selvittämään, kun käydään läpi kaikki kentän ruudut, ja katsotaan, mitä ominaisuuksia ruuduilla on. Jos ruudulla on esimerkiksi start-ominaisuuden arvo true, voidaan kyseisen ruudun koordinaatit ottaa talteen ja käyttää myöhemmin reitinetsinnässä. Reitinetsinnän ja ruutujen ominaisuuksien avulla voidaan toteuttaa paljon pelin toiminnallisuuksia. Esimerkiksi pelin lopettaminen, kun tarpeeksi monta vihollista on päässyt loppuun, toteutettiin reitinetsinnän kanssa. Kun vihollinen pääsee lopetusruudun päälle, reitinetsintä huomaa sen, ja haluttu toiminto, kuten pelin lopettaminen voidaan suorittaa.

AndEngine ja Tiled Map Editor toimivat tässä projektissa erittäin hyvin. Saimme toteutettua kaikki haluamamme ominaisuudet niiden avulla. AndEnginen dokumentaation puuttuminen oli tosin melko suuri haitta, varsinkin näin ensimmäisessä peliprojektissa. Melkein kaikki tarvittava tieto löytyi netistä etsimällä, ja itse kokeilemalla, mutta dokumentaatio olisi auttanut todella paljon. Tiled Map Editorista en löydä mitään moitittavaa. Se on tällaiseen projektiin juuri sopivan yksinkertainen työkalu, jolla on helppo päästä alkuun. Ruuduille asetettavat ominaisuudet ovat todella hyödyllisiä haluttujen toimintojen toteuttamisessa.

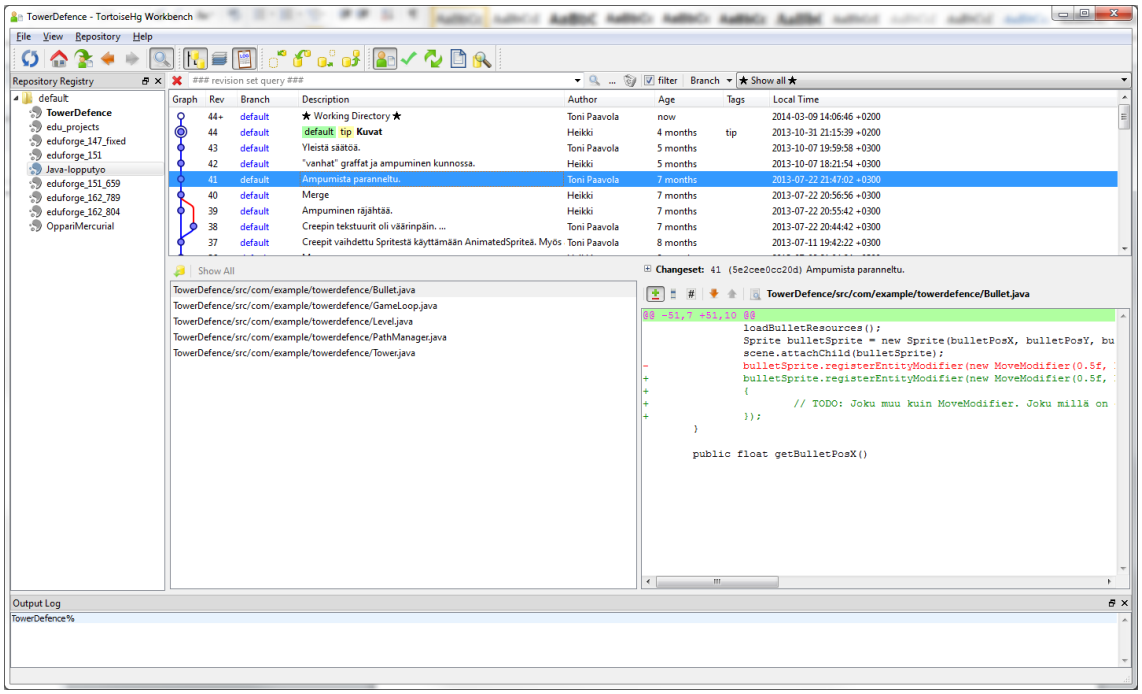
7.3.4 Mercurial, Bitbucket ja TortoiseHg

Mercurial valikoitui versionhallintajärjestelmäksi kokemuksen perusteella. Siitä on vaikea sanoa hyviä tai huonoja puolia, sillä mielestäni se toimi tässäkin projektissa, niin kuin versionhallinnan kuuluukin toimia. Minulla ei ole kokemusta muista versionhallintajärjestelmistä, joten niihin vertailu on mahdotonta. Mielestäni versionhallinta on loistava apuväline lähes missä tahansa projektissa ja siksi käytin versionhallintaa myös tämän raportin tekemisessä.

Ilman versionhallintaa, olisi opinnäytetyön tekeminen etätyöskentelynä kahdestaan lähes mahdotonta. Versionhallinta mahdollisti sekä minun että Heikin työskentelyn pelin parissa samaan aikaan tai erikseen. Omat muutokset saa lähetettyä Bitbucketissa sijaitsevaan tietovarastoon, josta toinen osapuoli saa ladattua ne itselleen ja jatkettua kehittämistä. Jos olimme muokanneet samoja tiedostoja, versionhallintajärjestelmällä muutokset sai yhdistettyä ilman ongelmia. Ilman versionhallintaa samaan tiedostoon tehtyjen muutosten yhdistäminen olisi ollut todella työlästä.

Bitbucketin käyttö oli hyvin yksinkertaista, mikä johtui osittain aikaisemmasta kokemuksesta, mutta myös Heikki, jolla aikaisempaa kokemusta ei ollut, oppi sen käytön nopeasti. Heikki loi tunnukset palveluun, jonka jälkeen hän pystytti uuden tietovaraston. Hän antoi minulle käyttöoikeudet tietovarastoon, jonka jälkeen me molemmat kloonasimme sen omalle koneellemme. Kloonaamisen jälkeen meidän ei tarvinnut välittää Bitbucketin toiminnasta, vaan kaikki versionhallintaan liittyvä tapahtui omilla koneillamme.

Mercurial ja Bitbucket toimivat siis ikään kuin taustalla, ja varsinainen versionhallinnan käyttö tapahtui meidän tapauksessamme TortoiseHg-työkalupaketin graafisilla työkaluilla. Olin käyttänyt versionhallintajärjestelmiä aiemmin komentoriviltä vain hieman koulussa ja ensimmäisen kerran graafisia työkaluja kokeiltuani tulin siihen tulokseen, että niiden käyttäminen on minulle paljon helpompaa. Tässä projektissa käytimme TortoiseHg Workbench -ohjelmaa (kuva 8).



Kuva 8. Perusnäkymä TortoiseHg Workbench -ohjelmassa.

Edellä olevassa kuvassa on nähtävillä Workbench-ohjelman perusnäkymä. Ylhäällä rivissä olevilla kuvakkeilla voi käyttää versionhallinnan keskeisimpiä komentoja kuten: pull, push, ja commit. Vasemmalla olevassa ikkunassa näkyvät kaikki tietovarastot, jotka on liitetty Workbench-ohjelmaan. Ylimmässä ikkunassa näkyy lista kaikista muutosjoukoista eli revisioista. Listasta voi valita hiirellä minkä tahansa revision, ja päivittää oman työhakemistonsa kyseiseen revisioon tai vain tarkastella, muutoksia ja lisäyksiä, joita projektin siinä vaiheessa on tehty. Viimeksi mainitun ikkunan alla on kaksi ikkunaa, joista vasemmanpuolimmaisessa näkyvät valitun revision muuttuneet, lisätyt tai poistetut tiedostot. Oikeanpuolimmaisessa ikkunassa näkyy valitun tiedoston muutokset ja ikkunan yläosassa näkyy myös kommit-viesti.

Workbench tarjoaa heti perusnäkymässään valtavan määrän tietoa tietovarastosta, ja eri revisioiden välillä siirtyminen on helppoa ja nopeaa. Revisioissa tehdyt muutokset saa näkymään vaivattomasti, ja samassa näkymässä näkyy myös tarkempaa tietoa varsinaisista tiedostoihin tehdyistä muutoksista. Kaikki tarpeellinen on korkeintaan parin hiiren painalluksen päässä. Graafisen työkalun käyttäminen on mielestäni todella paljon helpompaa, kuin komentorivillä komentojen kirjoittaminen, ja meidän käytössämme en jäänyt kaipaamaan Tor-

toiseHg Workbench -ohjelmalta enempää ominaisuuksia. Versionhallinnan käyttö sujui ongelmitta.

Ohjelmistoprojekteissa on mielestäni aina järkevää käyttää versionhallintaa, olivat ne sitten kuinka pieniä tai laajoja tahansa. Versionhallinnan tarve korostuu entisestään etätyöskentelyssä, jolloin ilman versionhallintaa toimiminen on käytännössä mahdotonta. Tässä projektissa käytetyt menetelmät toimivat hyvin ja mielestäni graafinen työkalu versionhallinnan käyttämiseen on ehdoton. TortoiseHg-työkalupaketissa on myös se hyvä puoli, että se sisältää kaiken, mitä Mercurial-versionhallinan käytön aloittamiseksi tarvitaan.

8 Pohdinta

Tässä luvussa vedän yhteen hieman sitä, miten koko opinnäytetyöprojekti onnistui. Käyn myös läpi miten mielestäni onnistuimme tower defence -pelin rungon tekemisessä ja pohdin miten valitsemamme työkalut lopulta sopivat käyttötarkoitukseemme ja olisimmeko voineet tehdä koko projektissa jotain paremmin.

8.1 Projektissa tehtyjen valintojen onnistuminen

Koko opinnäytetyön aiheiksi valitut mobiilipelin tekeminen ja tutustuminen mobiilikehitykseen olivat mielestäni onnistuneet valinnat. Mobiilipelaaminen on ehdottomasti ajankohtainen aihe, ja kokemus mobiilipelien tekemisestä ei varmasti ole haitaksi ohjelmoijalle tänä päivänä. Etätyöskentely on myös jatkuvasti kasvava työskentelymuoto. Varsinkin IT-alalla sen toteuttaminen on usein mahdollista helpommin kuin monilla muilla aloilla. Erilaisiin etätyötä helpottaviin apuvälineisiin tutustumisesta voi olla myös apua tulevaisuudessa.

Pariohjelmoinnin toteuttaminen etätyönä onnistui mielestäni mainiosti. Adobe Connect yhdessä Skypen käyttäminen viestinnässä mahdollistivat samanlaisen pariohjelmoinnin etänä, kuin mitä se on fyysisesti saman koneen ääressä. Molemmissa ohjelmissa oli pieniä puutteita, mutta ne täydensivät oivasti toisiaan.

Tällaisen peliprojektin toteutuksessa versionhallintajärjestelmän käyttö on käytännössä pakollista. Etätyössä versionhallinta korostuu entisestään, ja valitsemamme työkalut siihen toimivat moitteettomasti. Kuten viestintävälineidenkin kanssa, myös versiohallintajärjestelmä valikoitui lähinnä kokemuksen pohjalta. Tämä oli mielestäni hyvä valinta, sillä aihealue muuten oli entuudestaan tuntematon. Ainakin osittain tuttujen välineiden käyttö etätyön mahdollistamiseksi oli tarpeen, jotta pystyimme keskittymään uuden asian opetteluun. Projektin toteuttaminen olisi saattanut olla liian raskasta, mikäli kaikki työskentelytavat ja ohjelmat olisivat olleet tuntemattomia.

Kehitysympäristön kanssa onnistuimme mielestäni hyvin Eclipsen ja Android Developer Toolsin osalta. Tälle yhdistelmälle oli paljon ohjeita ja se tuntui olevan eniten käytetty vaihtoehto Android-kehityksessä ylipäätänsä. Eclipse oli entuudestaan tuttu, josta oli paljon apua. Pelimoottorin valinta olisi voinut olla näin jälkikäteen ajateltuna toinen. AndEnginessä sinänsä ei ole mitään vikaa, mutta dokumentoinnin puuttumisesta aiheutuneet ongelmat olivat joissain kohdissa melkoinen hidaste. Onneksi oppaita löytyi paljon, joten saimme ongelmat ratkaistua. Ehkä dokumentoinnin puuttumisessa oli myös omat hyvät puolensa. Tässä tapauksessa jouduimme toden teolla etsimään ratkaisuja ja paneutumaan AndEnginen toimintaan syvällisesti.

8.2 Tavoitteiden täyttyminen tower defence -pelin osalta

Pääsimme mielestäni tavoitteisiimme tower defence -pelin rungon tekemisessä. Saimme toteutettua halutut ominaisuudet ja peli jäi siihen tilaan, että jatkokehitys on mahdollista. Saimme toteutetuksi seuraavat ominaisuudet:

- Yksinkertainen valikkorakenne (päävalikko ja kentän valinta).

- Torneja pystyy lisäämään, ja ne ampuvat vihollisia.
- Viholliset liikkuvat haluttua reittiä pitkin alusta loppuun.
- Vihollisten liikkeiden animointi.
- Erilaisten vihollisten ja tornien tekeminen mahdollista.

Peli on vielä kaukana valmiista, mutta ainoastaan rungon tekeminen olikin tavoitteena. Jatkokehitykseen jäi vielä muun muassa seuraavat asiat:

- Enemmän kenttiä.
- Enemmän torni- ja vihollistyyppejä.
- Jonkin valuutan luominen, jolla torneja ostetaan ja päivitetään.
- Asteittain vaikeutuva vaikeusaste (kenttäsuunnittelu)
- Äänet
- Yleinen viimeistely

8.3 Päätelmä

Kaiken kaikkiaan olen todella tyytyväinen lopputulokseen, ottaen huomioon projektin lähtökohdat. Minulla tai Heikillä ei ollut kokemusta Android-pelikehityksestä ja pelimoottori oli myös entuudestaan tuntematon. Saimme vietyä projektin läpi etätyöskentelyn avulla, mikä ei lopulta eronnut hirveästi niin sanotusta normaalista työskentelystä. Jos lähtisin toteuttamaan uutta mobiilipeliprojektia, voisin käyttää melkein kaikkia samoja välineitä, kuin tässäkin projektissa. Pelimoottori olisi ehkäpä ainut, jonka vaihtaisin johonkin toiseen. Pelimoottorin dokumentaatio oli niin puutteellinen, että se vaikeutti työskentelyä melko paljon.

Lähteet

Aaltoyliopisto. 2011. Pariohjelmointi parantaa ohjelmistojen laatua teollisuusyrityksissä. http://www.aalto.fi/fi/current/news/view/2011-12-19-003/. 16.12.2013.

Adobe Help. 2013. Using Adobe Connect 8. http://help.adobe.com/en_US/connect/8.0/using/connect_8_help.pdf 6.1.2014.

Adobe. 2014. Visual quick start guide. http://www.adobe.com/content/dam/Adobe/en/products/adobeconnect/pdfs/VQS_Guide_for_Participants.pdf. 6.1.2014.

Andengine tutorials. 2014. Introduction to the AndEngine. http://www.matim-dev.com/introduction-to-the-andengine.html. 15.2.2014.

Android. 2014. http://www.android.com/versions/kit-kat-4-4/. 15.3.2014.

Androidsuomi.fi. 2014. Mikä on Android? http://blog.androidsuomi.fi/mika-on-android/ .19.12.2013.

Aniszczyk, C. & Gallardo,D. 2007a. Get started with the Eclipse Platform. http://www.ibm.com/developerworks/opensource/library/os-eclipse-platform/index.html. 10.2.2014.

Atlassian Bitbucket. 2014. Bitbucket Features . https://bitbucket.org/features. 19.01.2014.

Borho, S. TortoiseHg. 2014. http://tortoisehg.bitbucket.org/about.html 19.01.2014.

Bruner, N. 2012. Introduction to Tiled Map Editor: A Great, Platform-Agnostic Tool for Making Level Maps. http://gamedevelopment.tutsplus.com/tutorials/introduction-to-tiled-map-editor--gamedev-2838. 16.2.2014.

Cassavoy, L. 2013. What Makes a Smartphone Smart? http://cellphones.about.com/od/smartphonebasics/a/what_is_smart.htm. 21.12.2013.

Cisco. 2014. What is VoIP (Voice-over-IP)? http://www.cisco.com/en/US/prod/voicesw/networking_solutions_products_genericcontent0900aecd804f00ce.html. 6.1.2014.

Collins-Sussman, B., Fitzpatrick, B.W. & Pilato, M. C. 2011. Version Control with Sub version. http://svnbook.red-bean.com/en/1.7/svn-book.pdf 18.01.2014.

Developer Android. 2014. Android Developer Tools. http://developer.android.com/tools/help/adt.html#tools. 12.2.2014.

EngineersGarage. 2012. Android. http://www.engineersgarage.com/articles/what-is-android-introduction. 19.12.2013.

Git. 2014. Alkusanat - Versionhallinnasta. http://www.git-scm.com/book/fi/Alkusanat-Versionhallinnasta. 12.3.2014.

Hakala, P. 2013. Etätöitä ei tehdä vain kotona. http://www.hs.fi/talous/a1361944217099 15.12.2013.

Hildenbrand, J. 2013. Inside the different Android Versions. http://www.androidcentral.com/android-versions. 19.12.2013.

Java. 2014. Learn About Java Technology. https://www.java.com/en/about/. 10.2.2014.

Lomas, N. 2014. Android Took 79% Global Share Of Smartphones In 2013 — But Grew At Its Lowest Rate Yet, Says Analyst. 15.3.2014.

Mercurial. 2014a. MercurialEclipse. http://mercurial.selenic.com/wiki/MercurialEclipse. 18.1.2014.

Mercurial. 2014b. mercurial. http://mercurial.selenic.com/downloads/. 18.1.2014.

O'Sullivan. 2009. Mercurial: The Definitive Guide. http://hgbook.red-bean.com/read/a-tour-of-mercurial-the-basics.html. 19.1.2014.

OpenSignal. 2013. Android Fragmentation Visualized http://opensignal.com/reports/fragmentation-2013/. 19.12.2013.

Ricardo, D. 2013. AndEngine 1: Introduction to the AndEngine, Android video game libraries. https://www.ibm.com/developerworks/community/blogs/TamanKeet/entry/andengne_1_introduction_to_the_andengine_android_video game_libraries24?lang=en. 15.2.2014.

Saavalainen, H. 2013. Etätyö on tehokasta, mutta töistä on vaikea irrottautua. http://www.hs.fi/kotimaa/Etätyö+on+tehokasta+mutta+töistä+on+vaikea+irrottautua/a1379555211693. 15.12.2013.

Skype. 2014. Ominaisuudet. http://www.skype.com/fi/features/. 18.2.2014.

Tablet-PC. 2014. Tietoa tableteista. http://www.tablet-pc.fi/tablet/. 4.11.2.2013.

Techopedia. 2014. Software Library. http://www.techopedia.com/definition/3828/software-library. 15.2.2014.

Työ- ja elinkeinoministeriö. 2008. Etätyö. http://www.mol.fi/mol/fi/02_tyosuhteet_ja_lait/0161_etatyo/index.jsp 16.12.2013.

w3schools. 2014. XML Tutorial. http://www.w3schools.com/xml/. 12.2.2014.

Versionone. 2013. Pair Programming. http://www.versionone.com/Agile101/Pair_Programming.asp. 16.12.2013.

Victor, H. 2013. Android's Google Play beats App Store with over 1 million apps, now officially largest. http://www.phonearena.com/news/Androids-Google-Play-beats-App-Store-with-over-1-million-apps-now-officially-largest_id45680. 19.12.2013