Mobiilipelikehitys Android-älypuhelimeen AndEngine-pelimoottorin avulla

Matti Lähdeniemi

Opinnäytetyö
Toukokuu 2013
Tietotekniikka
Ohjelmistotekniikka

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

MATTI LÄHDENIEMI:
Mobiilipelikehitys Android-älypuhelimeen AndEngine-pelimoottorin avulla

Opinnäytetyö 46 sivua
Toukokuu 2013

Tämän opinnäytetyön tarkoituksena oli opettaa työnsä tekijälle Android-älypuhelimille suunnatun mobiilipelin toteuttamista, aina tarvittavien työkalujen omaksumisesta itse ohjelmointiin. Tähän pyrittiin toteuttamalla yksinkertainen pelisovellus AndEngine-pelimoottoria käyttäen.

Työssä tutustuttiin Android-laitteiden ominaisuuksiin ja niissä käytettävään käyttöjärjestelmään, ohjelmistokehitykseen ja itse sovelluksen toteutukseen, erityisesti valmiiseen pelimoottoriin AndEngineen keskittyen. Ohjelmistokehitys-osassa käsiteltiin kehitysympäristöä, hyödynnettävää pelimoottoria AndEngineä sekä sovelluksen testausta emulaattorilla ja laitteessa.

Esimerkkiohjelman suunnittelu- ja toteutuskappaleissa käsiteltiin toteutetun pelisovelluksen suunnittelua, käytettyjen ratkaisujen toimintakelpoisuutta sekä pohdittiin niiden toimivuutta ja mahdollisia jatkokehitysideoita, sovelluksen ehostamiseksi ja julkaisukelpoiseksi saattamiseksi.

Julkaisu-osassa käytiin lävitse toimenpiteitä, jotka tarvitaan valmistuvan sovelluksen julkaisemiseen Android-sisältöpalvelu Google Playssa.

Android-käyttöjärjestelmälle sovellusten ohjelmointi tapahtuu pääasiallisesti Java-ohjelmointikielellä. Tämän yleisesti käytetyn oliopohjaisen kielen lisäksi Android-sovelluksissa voidaan hyödyntää muun muassa XML-merkintäkieltä, jolla tehostetaan sovellusten toimintaa ja mahdollistetaan esimerkiksi tiedon välittäminen web-sivujen kanssa.

Työssä toteutettu esimerkkisovellus oli tarkoituksellisen yksinkertainen pelisovellus, jonka tarkoitus oli ennen kaikkea tutustuttaa työn tekijä pelisovellusten luomiseen yksinkertaisten rakenteiden ja ohjelmakutsujen avulla, varsinaisen valmiin myyntituotteen toteuttamisen sijaan. Tutustumalla yleisiin pelisovelluksen elementteihin sekä toimiviksi todettuihin toteutusratkaisuihin luodaan osaamispohjaa, jonka avulla tulevien peliprojektien toteuttaminen tulee olemaan helpompaa.

Asiasanat: Android, Eclipse, Java, pelimoottori, mobiilipeli

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Computer Science
Software Engineering

MATTI LÄHDENIEMI:
Mobile game development for Android smart phones using AndEngine game engine

Bachelor's thesis 46 pages
May 2013

The purpose of this thesis for the author was to learn about the process of developing a mobile game for Android smart phones, from adopting the necessary tools to the software development itself. This was pursued by implementing a simple demo game using the game engine AndEngine.

During the process the author researched Android devices and the operating system itself, application development for Android and creation of a mobile game using an already existing game engine. The section about software development was dedicated to the development environment and tools used during the process, to AndEngine game engine as well as to the testing of an application on the emulator and on an actual device.

The chapters about designing and implementing the demo game went through the designing of a game application, evaluating the competence of the solutions that were used as well as the effectiveness of them. In the latter part of the chapter potential ideas for further development and improvement of the demo application were considered, in order to generate a publishable form of the game in the future.

The chapter about the releasing process of an application went through the steps that need to be completed, in order to publish an application in the Google Play online store.

Applications for the Android operating system are developed with Java programming language. In addition to this commonly used object-oriented language, Android applications can be enhanced with the markup language XML, which can improve the application performance and enable sharing information with web pages.

The implemented demo game was deliberately a very simple game. Above all, the purpose of the implementation was to introduce to the author the creation of a game application through simple software structures and commands in the source code, instead of finishing a release candidate of a final product. By becoming familiar with the general game application elements, the primary goal of this thesis was to create a base of knowledge, for a future game project implementation to be easier to the author himself.

Key words: Android, Eclipse, Java, game engine, mobile game

SISÄLLYS

ERITYISSANASTO

Android	Googlen kehittämä, avoimeen lähdekoodiin perustuva mobiililaitteille suunnattu käyttöjärjestelmä.
Android SDK	Googlen tarjoama kehitystyökalupaketti, joka mahdollistaa ohjelmien tekemisen Android-käyttöjärjestelmälle.
avoin lähdekoodi	(engl. open source) Ohjelmistojen kehitysmenetelmät, jotka mahdollistavat käyttäjälle ohjelman lähdekoodiin tutustumisen ja sen muokkaamisen tarpeidensa mukaisesti. Lisäksi tarjoaa vapauden käyttää, kopioida ja levittää sekä alkuperäistä että muokattua versiota ohjelmakoodista.
.apk-paketti	Android application package file -tiedostotyyppi, johon pakataan Android-sovelluksen lähdekoodi sekä väliohjelmisto.
Google	Ohjelmistoyritys, joka on laajentanut toimialaansa hakukonepalveluista muun muassa Android-käyttöjärjestelmän kehitykseen ja Google Play -sisältöpalvelun ylläpitämiseen.
Eclipse IDE	Kehitysympäristö, jota yleisesti käytetään muun muassa Android-käyttöjärjestelmälle ohjelmoitavien sovellusten toteuttamiseen.
Java	Ohjelmointikieli, jota käytetään esimerkiksi Android-sovellusten kehittämiseen.
XML	Merkintäkieli, jota käytetään esimerkiksi Android-sovellusten kehittämiseen.
älypuhelin	Kehittynyt matkapuhelin, joka sisältää perinteisten puhelintoimintojen lisäksi kämmentietokoneiden ominaisuuksia. Jaotellaan usein käytettyjen käyttöjärjestelmien mukaan.
tablet-tietokone	Perinteisen kannettavan tietokoneen alalaji, jota pääasiassa käytetään kosketusnäyttönsä avulla.
AndEngine	Nicolas Gramlichin kehittämä, avoimeen lähdekoodiin perustuva pelimoottori Android-pelisovellusten kehittämiseen.

pelimoottori	Sovelluskehys, joka sisältää valmiita luokkia ja metodeja, joita yleisesti käytetään pelisovellusten kehittämisessä.
Google Play	Googlen ylläpitämä sisältöpalvelu.
middleware	Väliohjelmisto, joka toimii eräänlaisena välikerroksena sovelluksen ja käyttöjärjestelmän välillä, välittäen tietoa ja komentoja näiden välillä.
Dalvik	Prosessorivirtuaalikone, käytetään Android-käyttöjärjestelmässä esimerkiksi sovellusten suorittamiseen.
.dex-tiedosto	Dalvik Executable -tiedostotyyppi, johon pakataan käännetty Java-ohjelmakoodi prosessorivirtuaalikoneelle käyttö-kelpoiseen muotoon.
.class-tiedosto	Java-tavukoodin sisältävä tiedostotyyppi, jonka avulla Javan virtuaalikone voi suorittaa sovellukset.
debugger	Ohjelmakoodin toimintavirheiden paikantamiseen käytettävä työkalu, sisältyy usein valmiina osana kehitysympäristöön.
API	(engl. Application programming interface) Ohjelmointi-rajapinta, jonka mukaan ohjelmat voivat jakaa tietoja sekä komentoja keskenään.
emulaattori	Ohjelmallinen mallinnus esimerkiksi laitteesta tai käyttöjärjestelmästä, jonka avulla voidaan suorittaa kyseisen alustan sovelluksia ilman varsinaista natiivialustaa.
QEMU	Avoimen lähdekoodin prosessoriemulaattori, johon pohjautuva emulaattori sisältyy muun muassa Android SDK -pakettiin.
resoluutio	Näytön tarkkuus, joka ilmoitetaan yleensä kuvan piirtävien pikselien lukumääränä.
instanssi	Joko ohjelmallisen olion luokan tai esimerkiksi virtuaalikoneen yksittäinen edustaja tai ilmentymä.
GitHub	Git-versionhallintajärjestelmää hyödyntävä web-pohjainen hosting-palvelu, jota käytetään esimerkiksi ohjelmistoprojektien jakeluun.
ohjelmistokehys	Ohjelmistotuote, joka toimii varsinaisen sovelluksen runkona. Esimerkkinä pelisovelluksen perustoimintoja tarjoavat pelimoottorit.

first person shooter	(lyhennettynä FPS) Ensimmäisen persoonan ammuntapeli, yksi ensimmäisistä videopelityypeistä joissa hyödynnettiin pelimoottoreita.
porttaus	Ohjelman mukauttaminen toimimaan eri alustalla, jolle se on alun perin luotu.
.xml-tiedosto	Tiedostotyyppi, johon on tallennettuna esimerkiksi Android-sovelluksen käyttämä xml-merkintäkielinen toteutus.
kehitysympäristö	(engl. integrated development environment, IDE) Ohjelmistojen kehittämiseen suunniteltu työkalukokonaisuus, joka kattaa olennaiset työkalut ohjelmistojen kehittämiseen.
resoluutio	Näytön piirtotarkkuus, ilmoitetaan yleensä pikseleinä näytön leveys- ja korkeussuunnassa.
2D-peli	Kaksiulotteinen pelisovellus, joka sisältää pituus- ja leveysulottuvuudet.
ALV	Arvonlisävero.
USD	Yhdysvaltain dollari.
HUD	(Engl. heads-up display) Sovelluksen käyttöliittymässä näkyvät näytöt ja mittarit, joiden avulla välitetään sovelluksen käyttäjälle olennaisia tietoja: esimerkkinä pelisovelluksen HUD, jonka avulla pelaaja voi saada tietoja esimerkiksi pelin tavoitteista ja pelihahmonsa ominaisuuksista.

1 JOHDANTO

Tämän työn tarkoitus on tutustuttaa tekijänsä peliohjelmistojen kehitykseen Googlen Android-käyttöjärjestelmälle ja erityisesti älypuhelimille, joille kyseinen käyttöjärjestelmä on etupäässä suunnattu. Pääasiallisia käyttökohteita Androidille ovat useiden eri valmistajien tablet-tietokoneet ja älypuhelimet. Tämän työn ohjelmistokehityksen tarkastelu keskittyy valmista pelimoottoria hyödyntävän esimerkkipelisovelluksen luomisen vaiheisiin. Käytettäväksi pelimoottoriksi valikoitui AndEngine, ensisijaisesti Android-sovelluskehittämisen sekä pelimoottorin maksuttomuuden ansiosta.

Tässä työssä käsitellään Android-käyttöjärjestelmälle tarkoitettujen sovellusten kehittämiseen tarvittavia työkaluja ja sivutaan lyhyesti sovelluskehityksessä käytettyjä Java- ja XML-kieliä. Myös itse käyttöjärjestelmän ominaisuuksiin sekä kohdelaitteisiin luodaan yleistason katsaus.

Tarkemmassa tarkastelussa tutustutaan kehityksen ytimenä toimivan AndEnginen ominaisuuksiin sekä yleisemmin pelimoottorien rooliin pelikehityksen ydinkehyksenä. Tarkastelua syvennetään esimerkkiohjelman toteuttamisen avulla. Pääpaino luotavalla ohjelmalla on tarkastella yhdenlaisen ohjelmarakennemallin luomista valmiin pelimoottorin avulla ja tutustua peleissä yleisesti käytettyjen elementtien toteuttamiseen. Tehdyn sovellusluonnoksen tarkoitus onkin syventää ymmärrystä pelimoottorikehystä kohtaan, eikä luoda uraauurtavaa sovellusta.

Tämän työn luku 2 pitää sisällään kohdeympäristönä toimivan Android-käyttöjärjestelmän ja -laitteiden esittelyn. Luku 3 käsittelee Android-ympäristöön tapahtuvaa ohjelmistokehitystä ja siinä käytettäviä työkaluja. Luvussa 4 käsitellään käytettävän ohjelmistokehyksen, AndEngine-pelimoottorin ominaisuuksia sekä yleisellä tasolla pelimoottorien toimintaperiaatteita. Luku 5 esittelee toteutetun esimerkkiohjelman suunnittelua ja tavoitteita toteutuksen suhteen. Luvussa 6 käsitellään yleisellä tasolla esimerkkipelin toteutusta sekä toteutettujen ohjelmaluokkien sisältöä ja rakennetta. Luku 7 esittelee yleisellä tasolla vaiheet, jotka läpikäydään valmiin sovelluksen julkaisemiseksi sovelluskauppa Google Playssä. Luvussa 8 pohditaan työn tekoa ja tavoitteiden täyttymistä.

2 KOHDEYMPÄRISTÖ ANDROID-ÄLYPUHELIN

Tässä luvussa käydään lävitse yleisesti työn kohdeympäristönä toimivaa Android-käyttöjärjestelmää sekä sitä hyödyntäviä laitteita. Lisäksi perehdytään sovellusten sijoittumiseen Android-ympäristössä sekä sovellusten suorittamisen toteutukseen.

2.1 Android-käyttöjärjestelmä

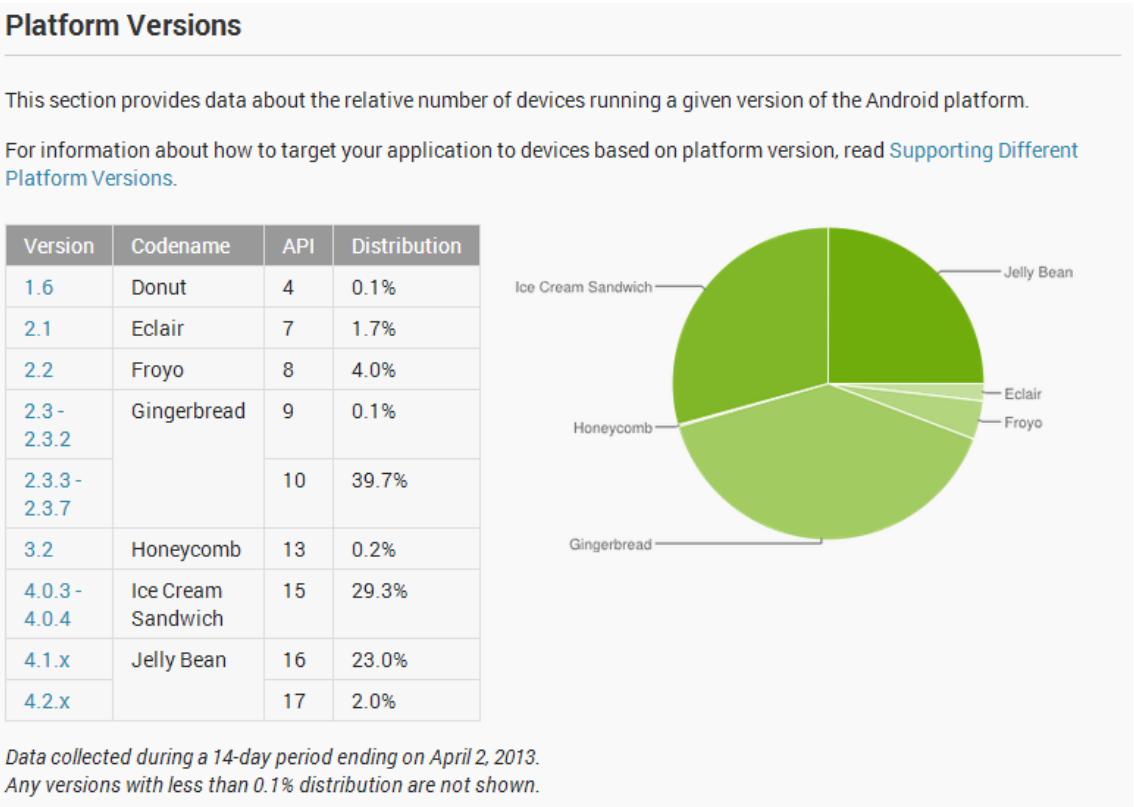
Android-käyttöjärjestelmä on alunperin vuonna 2003 perustetun Android Inc:n kehittämä käyttöjärjestelmä, joka kuuluu osana etupäässä mobiililaitteille suunnattua ohjelmistopinoa. Vuonna 2005 ohjelmistoyritys Google (joka aiemmin jo rahoitti Android Inc:n kehitystyötä) osti yrityksen tuotteineen. (Business Week: Google Buys Android for Its Mobile Arsenal)

Nykyisin käyttöjärjestelmän kehitystyöstä vastaa ohjelmistoyritysliitto Open Handset Alliance (perustettu vuonna 2007, julkistettiin samanaikaisesti Androidin kanssa). Ohjelmistoyritysliitto on usean suuren yrityksen konsortio. Yritysten toimialat vaihtelevat ohjelmistotuotannosta puolijohteiden ja mobiililaitteiden valmistamiseen. Merkittäviä jäseniä liitossa ovat Googlen lisäksi esimerkiksi puolijohdeteollisuuden toimijat Intel ja Nvidia sekä monialayritykset Samsung ja LG (edellä mainitut yritykset ovat myös liiton perustajajäseniä). (Open Handset Alliance: Overview, Open Handset Alliance: Industry Leaders Announce Open Platform for Mobile devices)

Käyttöjärjestelmä (kuten myös itse Android-ohjelmistopino) on osa Googlen johtamaa The Android Open Source Projectia, jonka päätehtävä on ylläpitää ja kehittää Androidia. Projektin päätarkoitus on rakentaa tavallisille käyttäjille ohjelmistoalusta, jonka lähdekoodi on avoinna muokkaamista ja esimerkiksi eri alustoille porttausta varten. (Android Open Source Project: About the Android Open Source Project)

The Android Open Source Project on pääasiallisesti Apache 2.0-lisenssin alaista ohjelmistoa: tämän lisenssin alaisuuteen kuuluvat muun muassa itse käyttöjärjestelmä sekä suurin osa siihen valmiina sisältyvistä ohjelmista. Ohjelmistopinoa koskeva lisensointi mahdollistaa joka tapauksessa projektin ja sen tuotteiden vapaa lähdekoodi -luonteen ja tämän tuoman vapauden kehitystyössä, niin laite- kuin sovellustuotannossa. (Open Handset Alliance: Open Handset Alliance Releases Android SDK)

Tämän työn kirjoittamishetkellä Android on helmikuussa 2013 julkaistussa vakaassa versiossa 4.2.2 (kutsumanimeltään ”Jelly Bean”). Googlen seuraaman Google Play -kirjautumishistorian perusteella yleisin käyttöjärjestelmäversio on kuitenkin edelleen Android 2.3 (kutsumanimeltään ”Gingerbread”), yli 45% osuudella kaikista Android-versioista. (Android Developer: Dashboards)



Platform Versions

This section provides data about the relative number of devices running a given version of the Android platform.

For information about how to target your application to devices based on platform version, read Supporting Different Platform Versions.

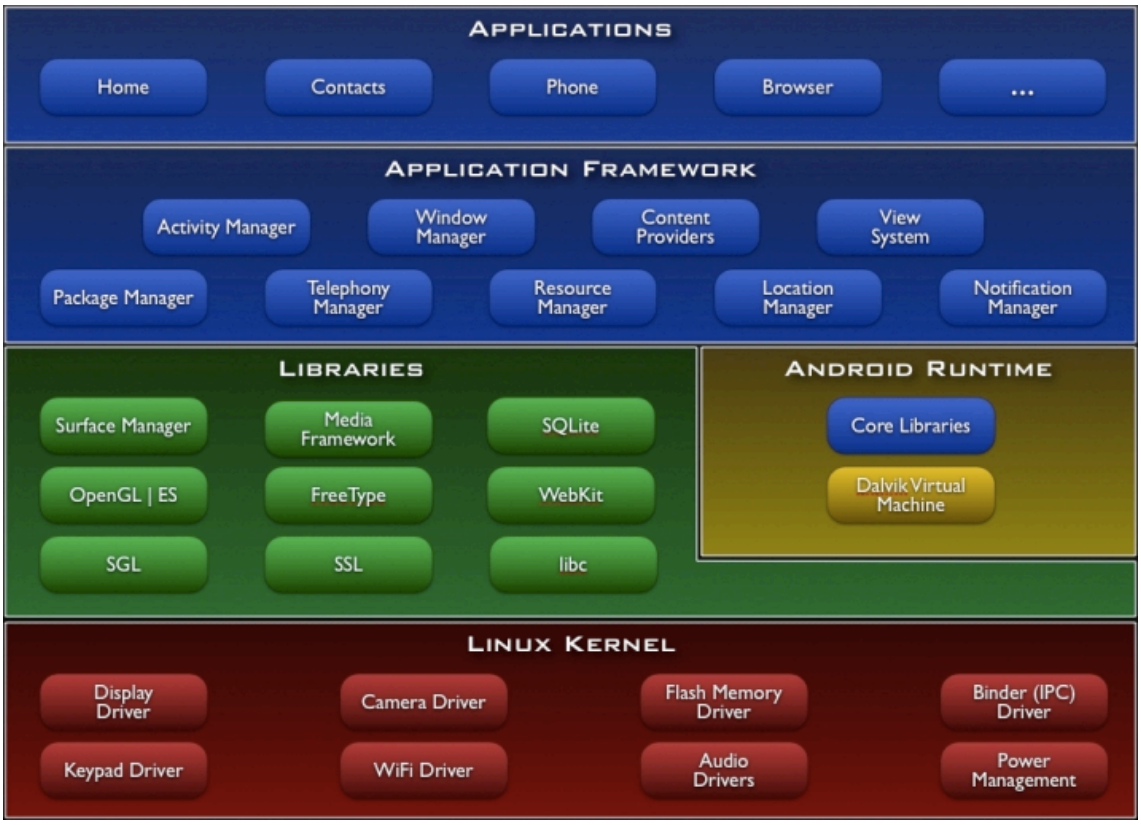
Version	Codename	API	Distribution
1.6	Donut	4	0.1%
2.1	Eclair	7	1.7%
2.2	Froyo	8	4.0%
2.3 - 2.3.2	Gingerbread	9	0.1%
2.3.3 - 2.3.7		10	39.7%
3.2	Honeycomb	13	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29.3%
4.1.x	Jelly Bean	16	23.0%
4.2.x		17	2.0%

Data collected during a 14-day period ending on April 2, 2013.
Any versions with less than 0.1% distribution are not shown.

KUVA 1. Kaavio käytössä olevien Android-versioiden jakaumasta huhtikuussa 2013. (Android Developer: Dashboards)

2.2. Android-käyttöjärjestelmän arkkitehtuuri

Alla näkyvä kaavio esittelee Android-käyttöjärjestelmän arkkitehtuuria:



KUVA 2. Kaavio Android-käyttöjärjestelmän arkkitehtuurista. (Android Developer: System Architecture)

Edellä esitellyn kaavion mukaisesti Android-arkkitehtuuri koostuu viidestä niin sanotusta pääelementistä, jotka asettuvat neljään tasoon. Ylin taso (ainoa suoraan järjestelmän käyttäjälle näkyvä) on tarkoitettu sovellusten ajamista varten: ajettavia sovelluksia ovat käyttöjärjestelmässä valmiiksi mukana tulevat sovellukset (kuten esimerkiksi kalenteri tai web-selain) sekä esimerkiksi Google Playn kautta julkaistut ja laitteeseen asennetut sovellukset. Tasoa alempana on sovelluskehys (Application Framework), joka tarjoaa rajapintoja sekä valmiita ohjelmistokomponentteja sovelluskehittäjille, kuten esimerkiksi sovellusaktiviteetteja hallinnoivan Activity Managerin. Sovelluskehys-taso päästää rajapintojensa avulla sovelluskehittäjät käsiksi tasoa alempana sijaitseviin kirjastoihin (pääasiassa C- ja C++ -kirjastoja). Kirjastojen rinnalle asettuu Android Runtime -kokonaisuus, joka sisältää Javan ydinkirjastojen lisäksi käyttöjärjestelmän sovellukset suorittavan Dalvik-prosessorivirtuaalikoneen. Arkkitehtuurin alimmalla tasolla on käyttöjärjestelmäydin Linux Kernel. Tasoa

käytetään muun muassa käyttöjärjestelmän turvallisuuteen ja prosessien hallintaan liittyviin toimintoihin. (Android Developer: App Framework)

2.3. Sovellusten ajaminen Android-käyttöjärjestelemässä

Androidissa käytettävät sovellukset ajetaan yksittäisinä prosesseina, joista jokaiselle käynnistetään oma Dalvik-virtuaalikoneen instanssi: tämä mahdollistaa useiden prosessien (täten myös sovellusten) yhtäaikaisen ajamisen. Ohjelmat luetaan ajonaikaisesti .dex-tiedostoista, jotka muunnetaan sovellusten Java-ohjelmakoodien tavumuunnokset sisältävistä .class-tiedostoista. Edellä mainitut .class-tiedostot puolestaan ovat Java-kääntäjän käännöksiä ohjelmoijan toimesta toteutetuista .java-lähdekooditiedostoista. (eLinux: Android Dalvik VM, Android Developers Blog: Dalvik JIT, Ehringer David: The Dalvik Virtual Machine Architecture)

Yksittäinen sovellus koostuu vähintään yhdestä aktiviteetista. Karkeasti ilmaistuna aktiviteetti on yksi käyttöliittymänäkymä, joka mahdollistaa sovelluksen käyttäjälle toimintojen käyttämisen (esimerkiksi sähköpostin kirjoittamisen). Sovellus voi koostua useastakin aktiviteetista, jolloin käynnistetyt aktiviteetit kasautuvat pinonomaisesti: siirryttäessä uuteen aktiviteettiin järjestelmä pysäyttää ja ”tallettaa” aikaisemman aktiviteetin pinonkaltaiseen muistiin. Uuden aktiviteetin sulkeutuessa, poistetaan tämä pinon päällimmäiseltä paikalta sekä kokonaan muistista ja siirrytään pinossa seuraavana esiintyvään, edellisenä avoinna olleeseen aktiviteettiin. (Android Developer: Activities)

2.4. Android-käyttöjärjestelmää käyttävät laitealustat

Avoimeen lähdekoodiin perustuva Android-käyttöjärjestelmä on houkutellut useita suuria(kin) valmistajia tuottamaan laitteita (pääasiassa älypuhelimia sekä tablet-tietokoneita), jotka hyödyntävät kyseistä järjestelmää. Tunnettuihin nimiin lukeutuvat muiden muassa Samsung, HTC sekä Motorola.

Laaja laitevalmistajien kirjo selittyy suurelta osin filosofialla, joka löytyy The Android Open Source Projectin taustalta: sen tarkoituksena on taata myös ohjelmistokehittäjille ja laitevalmistajille aina avoin vaihtoehto, jonka pohjalle luoda innovaatioita. Tähän filosofiaan tukeutuen on projektin tuotteen, Android-ohjelmistopinon (sekä täten myös käyttöjärjestelmän), uusin versio avoimessa jakelussa esimerkiksi projektin web-

sivuilla. Tämän lisäksi sivuilta löytyvät esimerkiksi vaatimusdokumentit, liittyen ohjelmiston (ja ohjelmistopinon) laitevaatimuksiin. (Android Open Source Project: Philosophy and Goals, Android Open Source Project: Compatibility Program Overview)

Avoimuudella on myös takaisin projektille lisäarvoa tuovia ominaisuuksia: laitevalmistajien antaessa palautetta sekä teknistä tukea takaisin Android Open Source Projectille, saa järjestelmän uudet kehitysversiot paljon suuntaa-antavaa tietoa aikasempien versioiden käyttökokemusten pohjalta. (Android Open Source Project, Android Open Source Project: Philosophy and Goals)

Sovellusten näkökulmasta laaja laitekanta asettaa myös omat haasteensa: erilaisten laitteiden suorituskykyjen sekä esimerkiksi näyttöresoluutioiden suuretkin vaihtelut tuottavat omat haasteensa sovellusten suunnitteluun sekä toteuttamiseen.

Vaikka Android onkin ennen kaikkea suunnattu niin sanotuille perinteisille mobiililaitteille (älypuhelimet sekä tabletit) ovat useat laitevalmistajat hyödyntäneet kyseistä käyttöjärjestelmää myös muiden innovaatioiden tukena: tämän työn kirjoittamishetkellä esimerkiksi laitevalmistaja Samsung on lanseerannut muun muassa perinteisen ”pokkarimallin” kameran sekä jääkaapin, joiden molempien toimintoja hyödynnetään Android-käyttöjärjestelmän avulla. Ajan saatossa tullaan vastaavia innovaatioita näkemään epäilemättä myös muissa arkisissa, uutta ratkaisua vaativissa tilanteissa, joiden haasteisiin mobiilikäyttöjärjestelmät soveltuvat vastaamaan. (The Telegraph: Samsung to launch Android-powered fridge)

Suurin haaste edellä mainittujen innovaatioiden toteuttamisessa on ollut järjestelmän luomat vaatimukset käyttölaitteiston suhteen: etenkin Androidin kaltainen kattava käyttöjärjestelmä asettaa tietyn vaatimustason sekä laitteiston muistille että suoritinkapasiteetille. Toisaalta, älylaitteiden osoittamat resurssit kannettavassakin koossa (esimerkkinä erittäin tehokkaat puhelimet sekä tabletit, niin muistin kuin prosessoritehonkin kannalta) sekä kyseisten teknologioiden yleistyminen tuovat käyttöjärjestelmät entistä lähemmäs myös näitä arkitilanteidenkin ratkaisuja.

3 OHJELMISTOKEHITYS ANDROID-LAITTEILLE JA KÄYTETTÄVÄT TYÖKALUT

Tässä luvussa esitellään lyhyesti Android-alustalle ohjelmoinnin mahdollistavat työkalut. Kappaleessa käsitellään lyhyesti myös Java-ohjelmointikieltä ja XML-merkintäkieltä sekä niiden roolia Android-sovelluskehityksessä.

Android-kehitykselle ja siihen tarvittaville työkaluille löytyy tuki ja omat ohjelmistoversionsa niin Windows-, Mac- kuin Linux-käyttöjärjestelmille. Tämän työn puitteissa keskitytään sovelluskehitykseen Windows-käyttöjärjestelmässä ja täten myös sen asettamiin vaatimuksiin.

3.1 JDK

Java Development Kit (lyhennettynä JDK) on Sun Microsystemsin kehittämä kehitystyökalupaketti, joka sisältää Java-ohjelmoinnissa vaadittavia kehitystyökaluja, muun muassa javac-kääntäjän sekä jdb-debuggerin. Olennainen osa JDK:ssa on myös sen sisältämä Java Runtime Environment (lyhennettynä JRE), ajoympäristö ja työkalukokonaisuus, joka mahdollistaa Javalla toteutettujen sovellusten suorittamisen. JDK on ilmaisessa jakelussa Oraclen verkkosivuilla: siellä ovat myös ajantasaiset asennus- ja päivitysohjeet paketille. (Oracle Technology Network: Java)

Ilman tarkempaa JDK:n sisällön erittelyä voidaan todeta sen olevan yksi vaadittavista paketeista Java-ohjelmoinnissa ja siksi sen mainitseminen myös tässä työssä on välttämätöntä.

3.2 Android SDK

Android Software Development Kit (lyhennettynä Android SDK) on Android Open Source Projectin tarjoama ohjelmistokokonaisuus, joka koostuu ohjelmistokehityksen kannalta pakollisista työkaluista Android-sovellusten tekemiseksi. Tämä kehitystyökalukokonaisuus sisältää esimerkiksi debuggerin, API-kirjastoja, QEMU-pohjaisen avoimen lähdekoodin emulaattorin sekä dokumentteja ja lähdekoodiesimerkkejä. (Android Developer: Tools Help)

SDK:n sisältämät osat mahdollistavat jo itsessään ohjelmistokehityksen Androidille. Käytännössä SDK:n rinnalla käytetään kuitenkin aina kehitysympäristöä, joka tehostaa ja helpottaa ohjelmistojen kehitystyötä. Tässä työssä SDK:n ohella käytettiin Eclipse IDE -kehitysympäristöä, joka esitellään tarkemmin omassa kappaleessaan.

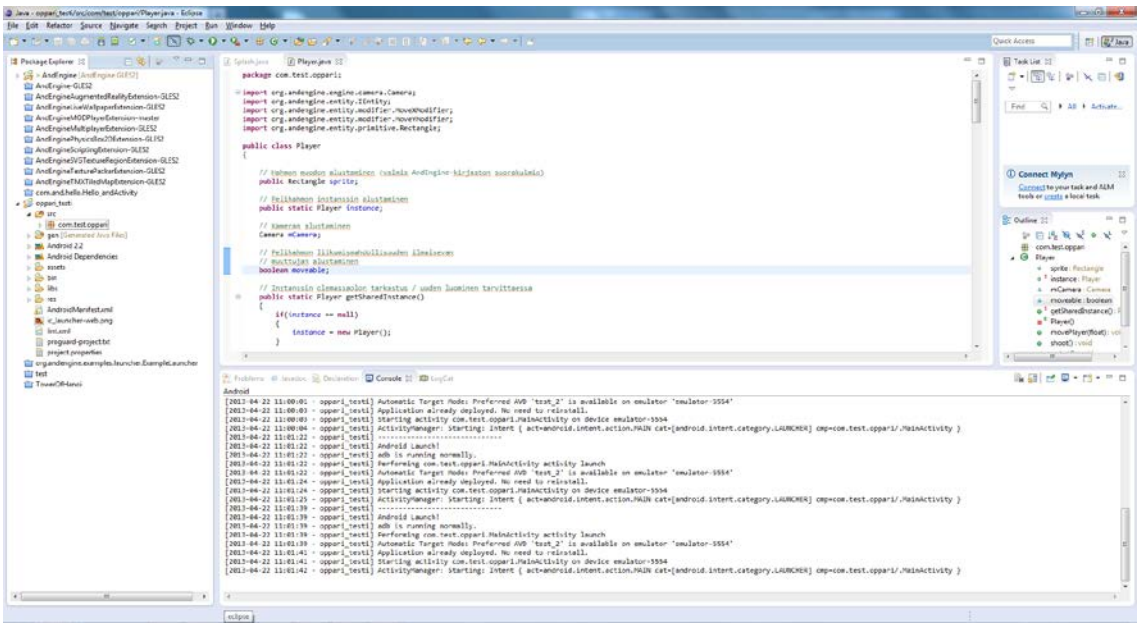
3.3. Kehitysympäristö Eclipse IDE

Eclipse Integrated Development Environment (lyhennettynä IDE) on alun perin IBM:n ja vuodesta 2004 lähtien Eclipse Foundationin kehittämä ohjelmistojen kehitysympäristö. Kehitysympäristö kuuluu osaksi Eclipse Foundationin projektia, jonka tavoitteena on tarjota avoin ohjelmistokehys sovellusten kehittämiseen ja hallintaan, niin yksilöiden kuin yritystenkin toimesta. (Eclipse Project: About the Eclipse Foundation)

Kehitysympäristö on yksi olennainen ohjelmistoprojektien työkalu. Usein sen peruselementtejä ovat integroidut käyttöliittymä ja ohjelmakoodin kääntäjä, joiden yhteentoimivuus helpottaa varsinaista kehitystyötä. Lisäksi kehitysympäristöön voi kuulua valmiina esimerkiksi debugger. Tässä työssä käytetty Eclipse IDE sisältää kaikki edellä mainitut ominaisuudet, muiden ominaisuuksiensa lisäksi. (Vogella.com: Eclipse IDE Tutorial)

Eclipse IDE on alustariippumaton ja Javan lisäksi sille löytyy tuetut versiot muun muassa C- ja C++ -ohjelmointikielillä kehitystä varten. Kaikki viralliset jakelupaketit ovat saatavissa Eclipse Foudationin web-sivuilla, josta löytyvät myös vaiheittaiset ohjeet jokaisen jakelun asentamiseksi ja ympäristön pystyttämiseksi. (Eclipse Project: Downloads)

Tämän työn kirjoitushetkellä Eclipse IDE varustettuna Android SDK:lla sekä ADT-pluginilla on Androidin kehittäjäyhteisön ainoa virallisesti tukema ilmainen kehitysympäristökokonaisuus. (Android Developer: Developer Tools)



KUVA 3. Kuvakaappaus Eclipse IDE -kehitysympäristöstä ohjelmointinäkymässä.

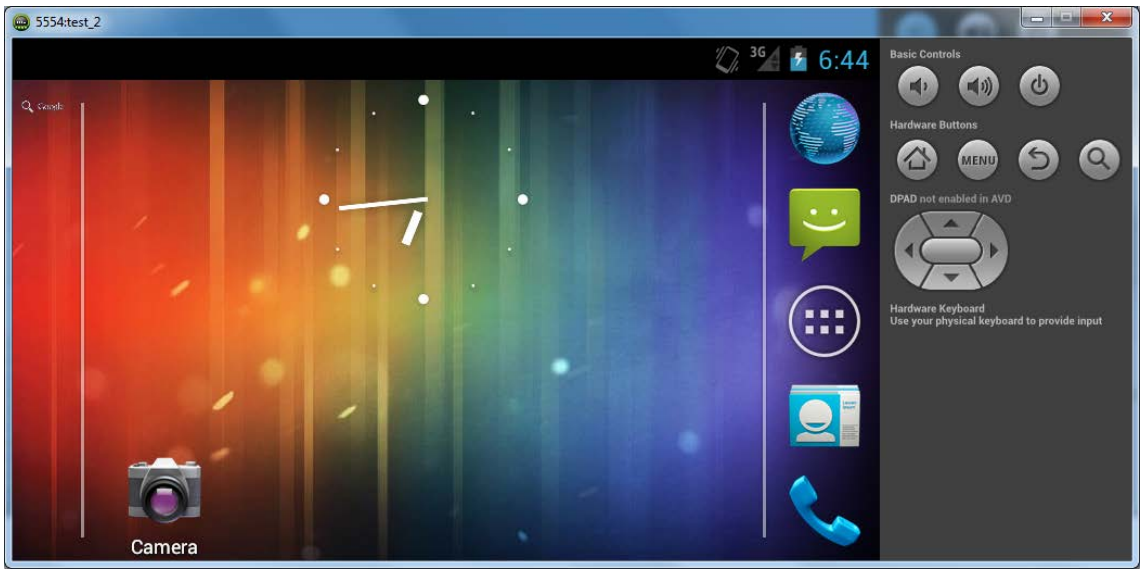
3.4. ADT Bundle

Android Development Tools (lyhennettynä ADT) Bundle on Androidin kehittäjäsivustolla tarjolla oleva ohjelmistopaketti, jonka lataamalla ja asentamalla voidaan ”suoraan” ohjelmoida sovelluksia Androidille. Bundle-paketti koostuu muun muassa edellisissä kappaleissa esitellyistä Android SDK:sta sekä Eclipse IDE:n jakelusta, johon on valmiiksi integroitu ADT-plugin. (Android Developer: Get the Android SDK)

Androidin kehittäjäsivustolla on paketin latausmahdollisuuden lisäksi tarjolla käyttöjärjestelmäkohtaisesti vaiheittaiset asennusohjeet sekä laitteistovaatimuksille Bundlelle. (Android Developer: Get the Android SDK)

3.5. Emulointi Eclipse-kehitysympäristössä

Eclipse IDE:n avulla on mahdollista luoda virtuaalisia Android-laitteita, joiden avulla Android-käyttöjärjestelmään ohjelmoituja sovelluksia voidaan testata ”todellisessa käyttöympäristössä” ilman varsinaista laitetta. Emuloitu Android-laite luodaan Eclipseen integroidulla Android Virtual Device (lyhennettynä AVD) Managerilla. (Android Developer: Managing Virtual Devices, Android Developer: AVD Manager)



KUVA 4. Kuvakaappaus AVD Managerilla luodusta emuloidusta Android-laitteesta.

Fyysisiin laitteisiin verrattaessa emuloinnin etuihin voidaan laskea ennen kaikkea muokkautuvuus: sovellusten testaus Android-käyttöjärjestelmän eri versioissa, erilaisilla laitekohtaisilla resoluutioilla onnistuu emuloiduilla laitteilla vaivattomasti ilman suurta laitekantaa. AVD Manager tukee myös useita tallennettuja virtuaalilaitteita: tämän ansiosta luodut emulaattori-laitteet voidaan tallentaa ja ottaa uudestaan käyttöön tarvittaessa.

3.6 Sovellusten testaus Android-laitteella

Eclipse IDE:lla käännetty ohjelmakoodi tuotetaan välittömästi .apk-paketiksi, jonka lataamalla Android-laitteeseen voidaan sovellus asentaa ja suorittaa oikeasti. Eclipse IDE:n avulla tämä siirtäminen onnistuu ohjatusti, jonka jälkeen paketti voidaan asentaa ja suorittaa laitteessa. (Android Developer: Using Hardware Devices)

Tämän työn puitteissa luotua esimerkkiohjelmaa testattiin ainoastaan emuloiduilla Android-laitteilla, mutta mahdollisuus oikeilla laitteilla testaamiseen on tärkeää mainita: esimerkiksi sovelluksia, joissa hyödynnetään useista Android-laitteista löytyvää, liikkeet ja kallistukset tunnistavaa kiihtyvyysmittaria voidaan testata ainoastaan oikeilla laitteilla.

3.7. Ohjelmointikieli Java

Java on Sun Microsystemsin kehittämä oliopohjainen ohjelmointikieli. Vuodesta 1991 alkaen kehityksessä ollut kieli on perustettu C++ -ohjelmointikielen pohjalta: tämä selittää kielten lähes identtisen syntaksin. Java on tulkattava ohjelmointikieli, joka käännetään virtuaalikoneen ajettavaksi, tämän ymmärtämään .dex-tiedostomuotoon (kuten Sovellusten ajaminen Android-käyttöjärjestelemässä -kappaleessa todettiin). (Oracle Technology Network: The History of Java Technology, IBM developerWorks: New to Java programming)

Koska Java-ohjelmat ajetaan virtuaalikoneen ”suljetussa ympäristössä”, ovat ne pääosin muilla konekielillä toteutettuja ohjelmia turvallisempia suorittaa. Kääntöpuolena virtuaalikoneella ajamisessa on sovelluksen suorituskyvyn lasku, jota Javassa on paikattu muun muassa muistihallintaa helpottavalla automaattisella roskienkeruu -menetelmällä. (Web-Dot-Dev: Java Advantages and Disadvantages, IBM developerWorks: Java theory and practice: A brief history of garbage collection)

Java mielletään helpoksi ohjelmointikieleksi oppia (jollaiseksi se on alun perin suunniteltukin) ja soveltuessaan monenlaisten ohjelmistojen tuottamiseen on se kasvattanut tasaisesti suosiotaan perustamisestaan lähtien. Esimerkiksi erilaisten web-sovellusten tekemiseen soveltuvuuden lisäksi Java on ensisijainen ohjelmointikieli Android-sovellusten tekemisessä. Sovellusten tekemistä voidaan tukea esimerkiksi XML-merkintäkielen avulla, joka on esitelty tarkemmin omassa kappaleessaan.

3.8. Merkintäkieli XML

Extensible Markup Language (lyhennettynä XML) on merkintäkieli ja WWW:n standardi, jonka kehityksen takana on World Wide Web Consortium: organisaatio ylläpitää ja kehittää myös muita WWW:n standardeja. Merkintäkielen avulla voidaan esimerkiksi jäsentää erilaisia tietokokonaisuuksia selkeämmiksi. Lisäksi sen avulla voidaan esimerkiksi tallentaa dokumentteja ja välittää tietoa erilaisten järjestelmien välillä. (W3C: Standards, W3C: XML Essentials)

Tämän työn puitteissa tehdyssä sovelluksessa XML-kieltä käytettiin ainoastaan sovelluksessa esiintyvien tekstien tallentamiseen, mutta hyödyntämistavat ovat

mobiilipelikehityksessäkin huomattavasti laajemmat. Tämänkin työn esimerkkisovelluksen jatkokehityksessä XML-kieltä voitaisiin hyödyntää esimerkiksi useiden pelikenttien toteutuksessa: kirjaamalla pelikentän elementit sijainteineen ja muine ominaisuuksineen kenttäkohtaisiin .xml-tiedostoihin, voitaisiin pelinäkymät ladata kenttäkohtaisesti halutun .xml-tiedoston tietojen pohjalta.

4 PELIMOOTTORI SOVELLUSKEHITYKSESSÄ, ANDENGINE-PELIMOOTTORI

Tässä luvussa käsitellään yleisellä tasolla pelimoottorin merkitystä pelisovelluksen pohjarakenteena ja käydään lävitse pelimoottorien erilaisia ilmenemismuotoja. Lisäksi käsitellään yksityiskohtaisemmin tämän työn esimerkkipelin kehityksen pohjana ollutta AndEngine-pelimoottoria ja sen yleisiä ominaisuuksia.

4.1 Pelimoottoreista yleisesti

Pelimoottori on ohjelmistokokonaisuus, joka on tarkoitettu videopelien kehittämiseen. Moottorin rooli sovelluksen suorittamisessa vaihtelee suuresti, tilanne- ja moottorikohtaisesti.

Yleensä pelimoottorin tärkein tehtävä on yleisen pelimekaniikan kattaminen, esimerkiksi peliobjektien graafinen mallintaminen ja näytölle piirtäminen. Pelimekaniikan lisäksi moottori voi sisältää pelisovelluksen hyödyntämiä fysiikkamallinnoksia tai tekoälyyn liittyviä toimintoja. Laajimmillaan pelimoottori voi olla kokonainen ohjelmistokehys, joka kattaa käytännössä kaikki elementit kokonaisen pelisovelluksen tekemiseksi. (Game Career Guide: What is a Game Engine?)

Varsinaisten pelimoottorien kehittäminen alkoi 1990-luvun puolivälissä, kun erilaiset 3D-pelit yleistyivät nopeasti. Pelimoottoripohjaisten pelien toteutusten kasvuun vaikuttivat etenkin first person shooter -pelit Doom ja Quake, joiden sovellusytiminä toimineet ”moottoriprototyypit” lisensoitiin ja niitä käytettiin muiden vastaavien pelisovellusten pohjana. (Game Career Guide: What is a Game Engine?)

Nykyisin ja tulevaisuudessa pelimoottorien käyttökohteet laajenevat perinteisestä viihdeteollisuudesta lukemattomiin suuntiin: niin sanottu Serious game -peligenre on yksi suurista ohjelmistoalan tuotehaaroista, jotka tulevat hyödyntämään pelimoottoreita tai vastaavia ohjelmistokokonaisuuksia omien sovellustensa toteuttamisessa. Serious game -genreä edustavat esimerkiksi erilaiset mallinnusohjelmat ja simulaattorit tai vaikkapa fysioterapeuttiseen kuntouttamiseen liittyvät lääketieteelliset sovellukset. (Gamasutra: The Designer’s Notebook: Sorting Out the Genre Muddle)

Edellä mainittujen ”vakavien pelien” lisäksi viihdekäyttöön tuotettavia pelejä toteutetaan laajalle laitekirjolle, mutta enenevässä määrin viihdesovelluksia tuotetaan varsinkin mobiililaitteille. Kevyempien pelisovellusten toteuttaminen mobiililaitteille on tuottanut alalle tulleiden sovelluskehittäjien lisäksi myös useita valmiita pelimoottoreita, joista monet ovat ilmaisia ja vapaasti käytettäviä: tällaisiin pelimoottoreihin lukeutuu myös tässä työssä käytetty AndEngine, joka on esitelty tarkemmin omassa kappaleessaan.

4.2. AndEngine-pelimoottori

AndEngine on saksalaissyntyisen ohjelmistokehittäjä Nicolas Gramlichin kehittämä pelimoottori, joka on suunnattu Android-käyttöjärjestelmälle kehitettävien 2D-pelien toteuttamiseen. Se hyödyntää OpenGL ES -grafiikkarajapintaa, joka on Khronos Groupin kehittämä avoin 3D-grafiikan ohjelmointirajapinta kannettaville laitteille. (GitHub:AndEngine / Readme, Khronos: OpenGL ES)

Keväästä 2010 lähtien kehityksessä ollut pelimoottori on sittemmin jakautunut kahdeksi erilliseksi kehityshaaraksi. Tätä työtä kirjoitettaessa näistä kehityshaaroista uudempi, grafiikkarajapinta OpenGL ES 2:a tukeva versio on enää jatkuvassa kehityksessä. Myös vanhempi, OpenGL ES 1:ä tukeva versio moottorista on edelleen jaossa. (GitHub:AndEngine / Readme)

Varsinainen AndEngine-moottori on oma middlewareksi miellettävä kirjasto, joka sisältää valmiita luokkia ja funktioita Android-ohjelmistopinon kirjastotasolle käsiksi pääsyä varten. Lisäksi AndEngine sisältää kattavasti valmiiksi toteutettuja, moottorin sisäisiä luokkia pelien tärkeiden elementtien luomista varten: esimerkiksi valikkonäkymien luomiselle tai peliobjektien fysiisten liikkeiden tarkastukselle löytyvät omat luokkansa, vain muutamia ominaisuuksia lueteltaessa.

Käyttöön AndEngine-pelimoottori otetaan esimerkiksi tässä työssä käytetyn Eclipse IDE:n avulla noutamalla se kehitysympäristön pakettienhallintaan ja liittämällä se oman peliprojektin kirjastoksi: tämä kirjaston valintamahdollisuus löytyy oman ohjelmistoprojektin ominaisuudet -välilehdeltä.

Alla oleva koodiesimerkki esittelee AndEnginen RatioResolutionPolicy-luokassa toteutetun onMeasure()-funktion. Kyseinen luokka mahdollistaa pelisovelluksen näkymien resoluution määrittämisen näkymäkohtaisesti. Luokan onMeasure()-funktiolla mahdollistetaan laitekohtainen näkymän resoluution määrittäminen: funktio noutaa Androidin View-luokan omistamat näytön mitat (leveys ja pituus) ja muokkaa niiden pohjalta pelisovelluksen kuvasuhteen mahdollisimman lähelle sovelluskehittäjän syöttämän ihannekuvasuhteen mukaista parametriarvoa:

```
@Override
public void onMeasure(final RenderSurfaceView pRenderSurfaceView,
final int pWidthMeasureSpec, final int pHeightMeasureSpec) {
            BaseResolutionPolicy.throwOnNotMeasureSpecEXACTLY
            (pWidthMeasureSpec, pHeightMeasureSpec);

            final int specWidth =
            MeasureSpec.getSize(pWidthMeasureSpec);
            final int specHeight =
            MeasureSpec.getSize(pHeightMeasureSpec);

            final float desiredRatio = this.mRatio;
            final float realRatio =
            (float)specWidth / specHeight;

            int measuredWidth;
            int measuredHeight;

            if(realRatio < desiredRatio) {
                        measuredWidth = specWidth;
                        measuredHeight =
                        Math.round(measuredWidth / desiredRatio);
            } else {
                        measuredHeight = specHeight;
                        measuredWidth =
                        Math.round(measuredHeight *
                        desiredRatio);
            }

            pRenderSurfaceView.setMeasuredDimensionProxy
            (measuredWidth, measuredHeight);
}
```

KUVA 5. Koodiesimerkki AndEngine-pelimoottorin RatioResolutionPolicy-luokasta. (Github: Andengine / RatioResolutionPolicy.java)

Edellä esitetty pelimoottorin ohjelmakoodipätkä on yksi esimerkki AndEnginen toiminnasta pelisovelluksen ja Androidin välisenä välittäjäohjelmistona. Vastaavia ratkaisuja löytyy moottorin perusversiostakin kymmeniä kappaleita, liittyen aina saatavilla olevan laitteiston tai sen resurssien paikantamisesta esimerkiksi sovelluksen orientaation muuttamiseen laitteen liikeanturien kuuntelun avulla. Tämän lisäksi

koodipätkä osoittaa suureksi haasteeksi osoittautuneen pelimoottorilta kokonaan puuttuvan kommentoinnin tai dokumentaation.

Suurin kynnys AndEngineä hyödyntävän sovelluksen toteuttamiselle syntyykin moottorin lähdekoodin kommentoimattomuudesta (kuten edellä esitelty koodiesimerkki osoitti). Varsinaisen kommentoinnin tai muunlaisen dokumentaation sijaan moottorin kehittäjä Nicolas Gramlich on luonut useita esimerkkisovelluksia, joissa hyödynnetään sekä moottoria että sen eri ominaisuuksia mahdollisimman monipuolisesti. Esimerkkisovellukset koodeineen ovat erittäin tarpeellisia ja hyödyllisiä, mutta nekään eivät poista sitä tarvetta, jonka varsinaisen moottorin ohjelmakoodin kommentoinnin puute aiheuttaa. Edellämainittujen esimerkkisovellusten lähdekoodit ovat avoimia ja tutustuminen moottorin käyttöön tapahtuukin ensisijaisesti näiden lähdekoodien avulla. (GitHub:AndEngine / Readme)

AndEnginelle ominaista on sen rakenne: varsinaisen ”perusmoottorin” lisäksi projekti sisältää laajennusosia, joista jokainen sisältää itselleen ominaisia lisäominaisuuksia varsinaisen moottorin käytettäväksi. Itse ”perusmoottori” sisältää olennaiset elementit ”peruspelin” toteuttamiseksi: sen ominaisuuksia ovat muun muassa peliluuppia pyörittävä moottori, sekä eri näkymiä hallinnoiva manageri-luokka. Esimerkiksi tässä työssä esimerkkisovelluksen toteutus sisältää ainoastaan perusmoottorin ja siten vain sen tarjoamat ominaisuudet. (GitHub:AndEngine / Readme)

Laajennusosia AndEngine-moottorille on tämän työn kirjoittamishetkellä 12 kappaletta: ne ovat omia lisäkirjastojaan, jotka liitetään tarvittaessa AndEnginen lisäksi oman projektin käyttöön: tämä tapahtuu Eclipse IDE:n kanssa työskennellessä samalla tavalla kuin perusmoottorinkin lisääminen. Erilaisiin laajennusosiin lukeutuvat muun muassa Box2D-laajennos: se hyödyntää nimensä mukaisesti avointa Box2D-fysiikkamoottoria, jota käytetään muun muassa Rovion menestystuote Angry Birdsissä. (GitHub:AndEngine / Readme)

Varsinaisen pelimoottorin dokumentaation jäädessä edellä mainittuihin esimerkkisovelluksiin, on pelimoottorin ympärille muodostunut vahva sovelluskehittäjien yhteisö, joka on koostanut erilaisia ohjelmaesimerkkejä sekä ongelmanratkaisumalleja pelimoottorin viralliselle keskustelupalstalle: vahva

yhteisöllinen tuki AndEngine-projektia kohtaan auttoi myös tämän työn tekemisessä, varsinkin esimerkkisovelluksen toteutuksen kanssa. (AndEngine: Forum)

Sekä perusmoottori että kaikki sen laajennososat ja esimerkkisovellukset ovat ladattavissa ilmaiseksi, ilman rekisteröitymispakkoa Nicolas Gramlichin omalta GitHub-tililtä. (GitHub: Nicolas Gramlich)

5 ESIMERKKIPELIN SUUNNITTELEMINEN

Tässä kappaleessa esitellään tämän työn esimerkkisovelluksena toteutetun pelisovelluksen suunnittelua. Työn tekijän tavoitteiden lisäksi käydään lävitse pelisovelluksen perusidea sekä sen tärkeimmät toiminnot.

5.1. Tavoitteet pelisovelluksen suhteen

Toteutetun pelisovelluksen tarkoitus oli ennen kaikkea tutustuttaa työn tekijä pelien tekemiseen. Tavoite valmiin tuotteen tekemisestä hylättiin yksinkertaisesti aikataulusyistä: valmiin pelisovelluksen toteuttaminen vie aikaa satoja, jopa tuhansia tunteja vaikka kyseessä olisi ainoastaan mobiilipeli. Pelkän pelilogiikan lisäksi valmis sovellus sisältää esimerkiksi grafiikoita ja äänitehosteita, joiden tekemiseen kuluva aika todettiin suosiolla parhaaksi jättää tämän työn ulkopuoliseksi toiminnaksi (etenkin näiden pelielementtien syntyessä ohjelmistokehittäjän itsensä toimesta).

Edellä mainitusta syystä esimerkkipelin toteutuksen tavoitteeksi asetettiin peleille olennaisten elementtien (käytännössä olennaiset näkymät siirtymineen ja toimiva pelinäkymä) sisällyttäminen toteutukseen: näin tavoite pelikehitykseen tutustumisesta saatiin toteutumaan työn aikataulun puitteissa.

Varsinaisen pelilogiikan osalta aihe valikoitiin tutun genren ja ominaisuuksien hahmottamisen helppouden kannalta: vastaavia sovelluksia ja muunnelmia klassisesta avaruus-ammuskelusta on avoimessa jakelussa kymmenittäin, ja erilaisten esimerkkitoteutusten seuraaminen ja ideoiden omaksuminen omaan toteutukseen helpotti logiikan luomista, jättäen käytetyn ajan mahdollisimman täysipainoisesti varsinaisen pelituotannon oppimiseen.

5.2. Ohjelman kuvaus ja idea

Toteutetun esimerkkipelin aiheeksi valikoitui muunnelma perinteisestä Space Invaders -kolikkopelistä. Pelityyppi edustaa klassista kaksiulotteista kolikkopeliä, joka sisältää useita perinteisiä videopelin elementtejä ja jonka toteuttaminen myös vähäisellä peliohjelmoinnin kokemuksella on mahdollista.

Tässä työssä toteutetussa pelin ”demo-versiossa” pelaajan tavoitteena on läpäistä yksi kenttä tuhoamalla kentällä olevat viholliset. Pelihahmo, joka liikkuu näytön alalaidassa vaakatasossa, voi tuhota yläpuolellaan leijuvat viholliset aseellaan.

5.3. Ohjelman toiminta ja ominaisuudet

Pelin demo-versiossa sovellus siirtyy käynnistyessään splash-näkymän kautta päävalikkoon. Pelaaja voi kokeilla pelin pelaamista päävalikon Aloita peli -painiketta painamalla tai poistua sovelluksesta Poistu pelistä -painikkeen avulla.

Varsinaisessa pelissä pelaaja pyrkii tuhoamaan viholliset, ohjaamalla pelihahmoa näytön laidoissa olevia ohjauspainikkeita painamalla ja ampumalla vihollisia pelihahmon aseella, jota käytetään muuta näyttöä painamalla.

Kaikkien vihollisten tuhouduttua pelinäyttöön ilmestyy onnitteluteksti. Tämän jälkeen pelinäkymä suljetaan näytön painalluksesta ja sovellus palaa takaisin pelin päävalikkoon.

6 ESIMERKKIPELIN TOTEUTUS

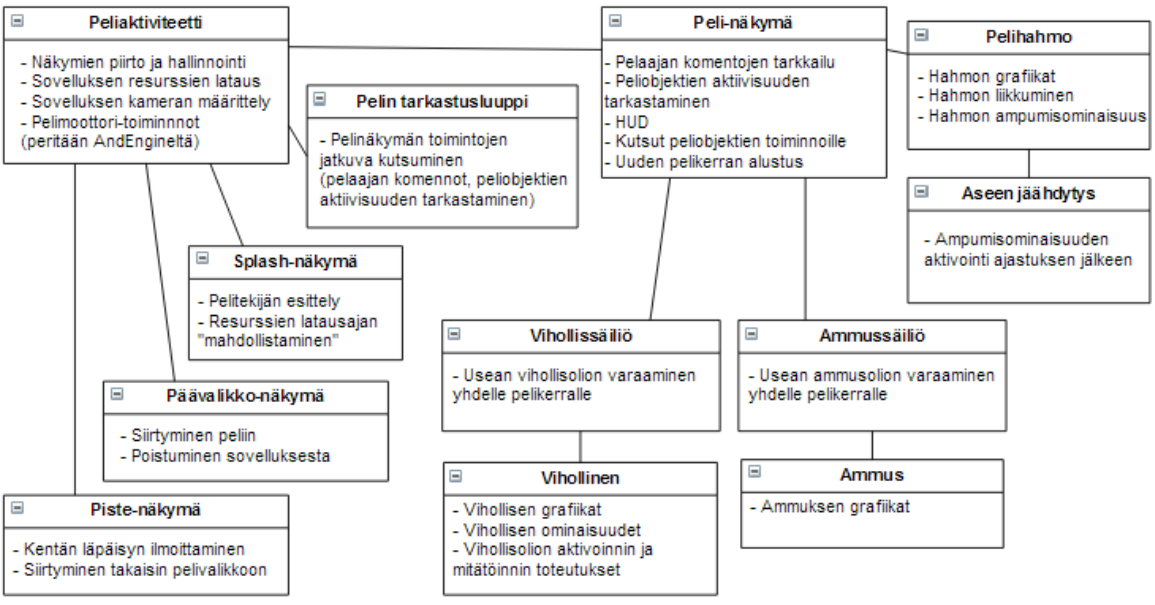
Tässä luvussa käydään läpi esimerkkisovelluksen toteutusta. Kaikki elementit on toteutettu AndEngine-pelimoottorin tarjoamilla toteutusvaihtoehdoilla, ellei toisin mainita. Teknisen kuvauksen lisäksi käsitellään toteutettujen ominaisuuksien roolia perinteisessä pelisovelluksen toteutuksessa.

Tämän esimerkkisovelluksen elementtien kuvauksen lisäksi luvussa pohditaan mahdollisia lisäominaisuuksia ja kehitysehdotuksia, joita jokaisen näkymän julkaisukelpoisessa versiossa voisi olla.

6.1. Ohjelman rakenne

Sovelluksen rakenne ja pelin eri elementtien toiminnot käsitellään vaiheittain, kahdessa erillisessä jaottelussa: ensimmäinen jaottelu käsittelee sovellusaktiviteetin toimintaa ja aktiviteetin käsittelemiä, pelin sisältämiä näkymiä. Toinen jaottelu pilkkoo varsinaisen pelinäkymän pienempiin osiin ja erittelee sen elementit.

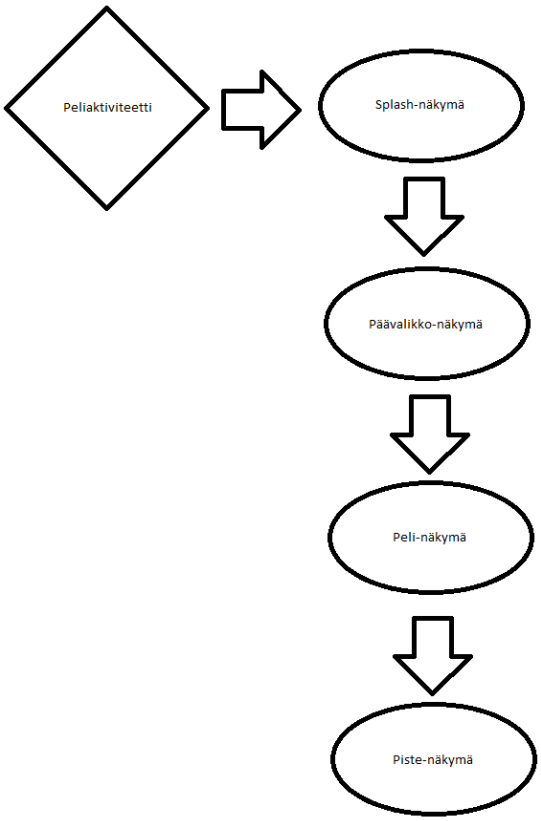
Alla esitetty kaavioluonnostelma esittelee pelin tämänhetkisen toteutuksen pääpiirteisen rakenteen luokkatasolla, esitellen demoversiossa toteutetut luokat ja näiden pääasialliset tehtävät:



KUVA 6. Esimerkkisovelluksen toteutetut luokat ja näiden pääasialliset tehtävät.

6.1.1 Yleinen rakenne, aktiviteetti ja sovelluksen eri näkymät

Alla olevassa kaaviossa esitellään karkeasti sovelluksen toimintaperiaate:



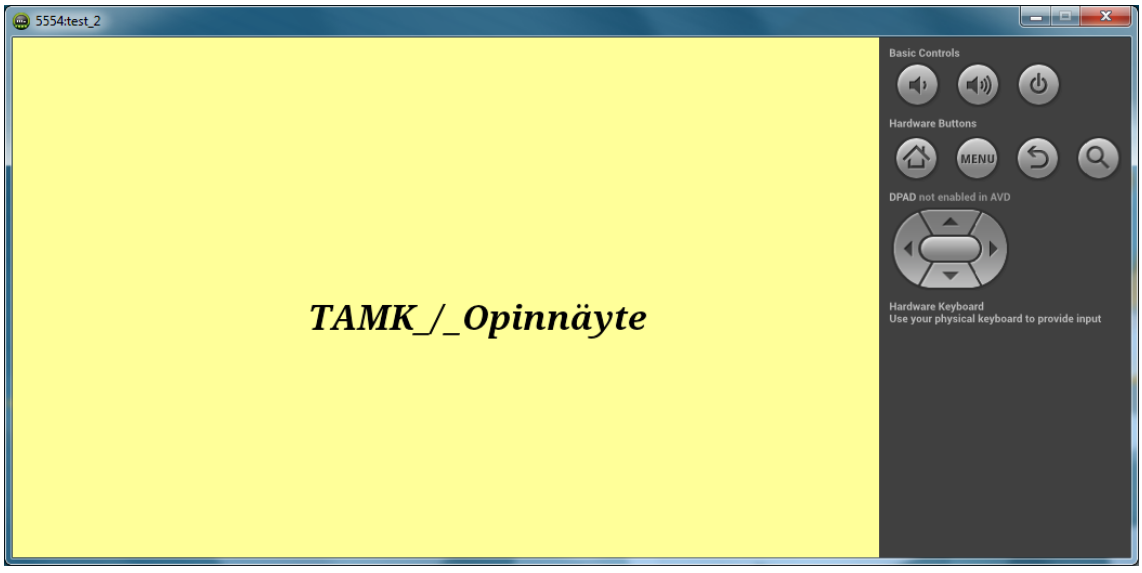
KUVA 7. Esimerkkisovelluksen siirtymäkaavio.

Toteutettu pelisovellus toimii yhdessä aktiviteetissa, joka käynnistetään sovellusta avattaessa. Aktiviteetti perii AndEnginen tarjoaman BaseGameActivity-luokan ominaisuudet: tämä valmis luokka pitää sisässään muun muassa sovelluksen varsinaisen moottorin, joka synkronoi erillisten näkymien piirtämistä sekä päivittämistä, jonka ansiosta sovellus ”etenee ja pyörii”. Näiden vaihtuvien näkymien hallinnointi sisältyy myös aktiviteettiin.

Edellä mainitun näkymien hallinnoinnin lisäksi aktiviteetissa määritetään sovelluksen käyttämä kamera resoluutioineen (korkeus ja leveys pikseleinä) ja ruutuorientaatioineen sekä sovelluksessa esiintyvien tekstien fontti. Lisäksi aktiviteetti lataa pelin käyttämät resurssit (tämän sovelluksen puitteissa funktio lataa sovelluksessa käytetyn fontin): tämä funktio ajetaan välittömästi ajastetun splash-näkymän käynnistämisen jälkeen.

Varsinaisia näkymiä, joiden näkyvyyttä aktiviteetissa vaihdellaan, sovellus sisältää neljä kappaletta: niihin lukeutuvat splash-ruutu, päävalikko, varsinainen pelinäkymä ja tämän päällä kentän läpäisyn jälkeen esitettävä pistenäkymä. Teknisesti jokainen näkymä on hyvin yksinkertainen ja toistensa kanssa yhdenmukainen: jokainen näkymä määrittää itselleen ominaisen taustakuvan ja toimintonsa. Siirtymissä jokainen näkymä asetetaan aktiviteetin näkyväksi näkymäksi, tuhoten edellisen näkymän käyttömuistin vapauttamiseksi.

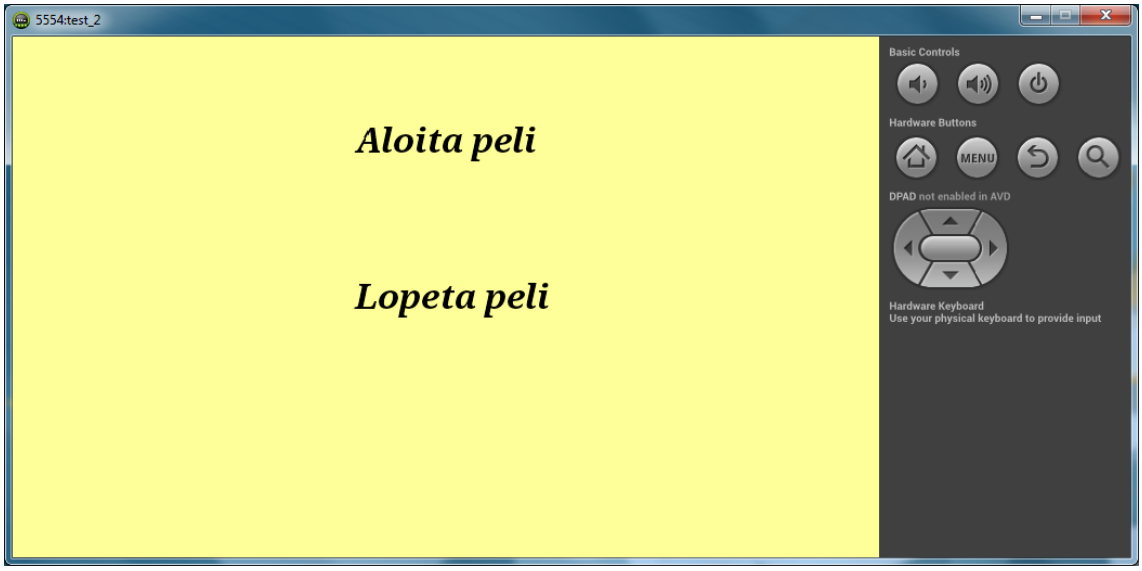
Sovelluksen käynnistyessä aktiviteetin näkymäksi asetetaan splash-ruutu, jonka olemassaololle on kaksi erillistä merkitystä. Perinteisesti splash-ruutu on ollut oivallinen paikka tekijän tai julkaisijan esittelylle, nimen tai logon muodossa. Tämän lisäksi splash-näkymän aikana sovelluksen vaatimat resurssit (kuten grafiikat ja äänitehosteet) saadaan ladattua sovelluksen käyttöön huomaamattomasti, ilman pelaajalle syntyvää ”erillistä odotteluaikaa”. Tässä toteutuksessa splash-ruutu on ajastettu, ja näyttöajan päätyttyä sovelluksen aktiviteetin näyttämäksi näkymäksi vaihdetaan päävalikko-näkymä, johon siirrytään automaattisesti. Splash-näkymässä näytetyt tekstit noudetaan ohjelmaprojektin resurssi-kansioon tallennetusta .xml-tiedostosta, joka sisältää myös muiden sovelluksessa käytettyjen string-muuttujien arvot (toisin sanoen sovelluksen muissakin näytöissä esiintyvät tekstit).



KUVA 8. Kuvakaappaus esimerkkisovelluksen splash-näkymästä emuloidussa Android-laitteessa.

Päävalikko asettaa itsensä aktiviteetin näkyväksi näkymäksi välittömästi splash-ruudun ajastetun näkyvyyden jälkeen. Se esittelee pelaajalle mahdolliset vaihtoehdot

sovelluksen sisällä: kokonaisessa pelisovelluksessa päävalikossa mahdollistetaan yleensä esimerkiksi vanhan pelitallennuksen lataaminen tai esimerkiksi pelissä käytettävien ääni- ja grafiikkatehosteiden asetusten muuttaminen. Tämän esimerkkisovelluksen toteutetussa päävalikkonäkymässä pelaajalla on kaksi etenemisvaihtoehtoa: painettaessa Aloita peli -painiketta sovellus käynnistää uuden pelin ja siirtää näkymän pelinäkymään, kun taas Lopeta peli -painiketta painettaessa sovellus suljetaan. Painikkeiden näkyvä osa on toteutettu string-muuttujilla, joiden arvot haetaan splash-ruudun tapaan ohjelmaprojektin resursseista, .xml-tiedostosta.



KUVA 9. Kuvakaappaus esimerkkisovelluksen päävalikko-näkymästä emuloidussa Android-laitteessa.

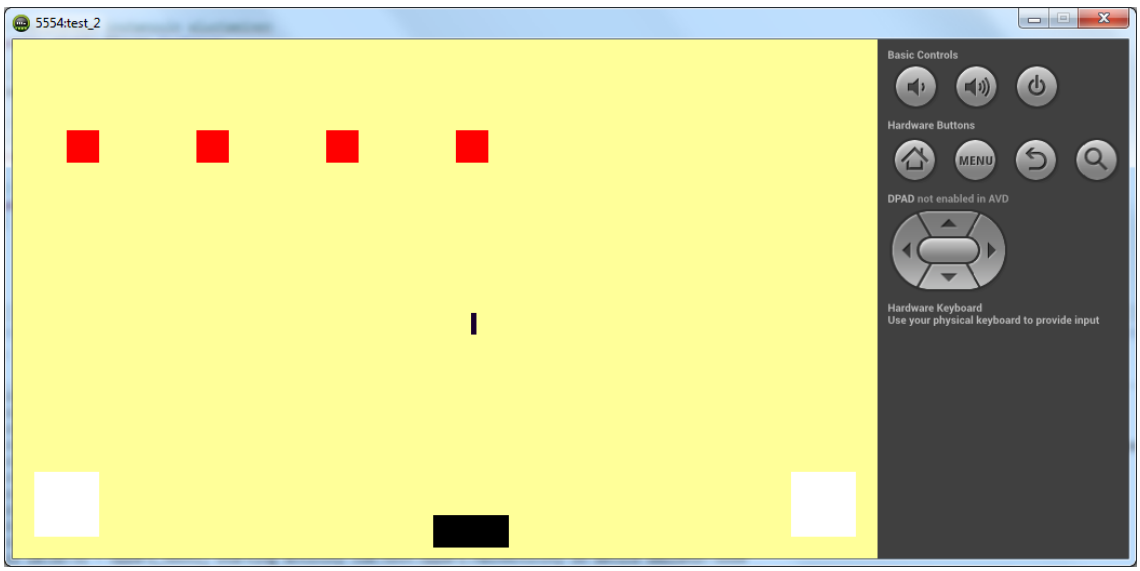
Varsinaisessa pelinäkymässä (joka käydään läpi tarkemmin omassa kappaleessaan) omana graafisena kerroksenaan esitetään pistenäkymä: tämän pistetaulun tarkoituksena on esittää esimerkiksi onnitteluteksti tai ansaitut pisteet, kun pelaaja on läpäissyt pelaamansa tason. Tämän sovelluksen toteutuksessa pistenäkymä siirretään näkymään pelinäkymän päälle, kun kaikki tason viholliset on tuhottu: se onnittelee pelaajaa onnistuneesta kentän läpäisystä tekstin muodossa, jonka arvo (siis näytetty teksti) haetaan ohjelmaprojektin resursseista, .xml-tiedostosta. Näkymästä poistutaan koskettamalla näyttöä: kosketuksesta suljetaan ja ”mitätöidään” myös käytetty pelinäkymä ja siirrytään takaisin päävalikko-näkymään.

6.1.2 Pelinäkymä ja sen elementit

Pelinäkymä asettaa itsensä aktiviteetin näkyväksi näkymäksi pelaajan painettua päävalikko-näkymässä Aloita peli -painiketta. Varsinaisen näkymän määrittelyn lisäksi pelinäkymä-luokka sisältää myös toiminnallisia ominaisuuksia: pelinäkymässä tapahtuvia painikkeiden painamisia tarkkaillaan ja painallukset tunnistettaessa kutsutaan toimintoja funktiot sisältävästä pelihahmo-luokasta. Lisäksi pelinäkymä-luokka sisältää tarkastus- ja puhdistusfunktiot näkymässä näkyville peli-, vihollis- sekä ammusolioille, joiden avulla jokainen aloitettu pelikerta alkaa ”uusilla” olioilla ja pelikerralla tuhotut oliot puhdistetaan näkymän käytöstä.

Esimerkkipelin lajiksi valikoitunut kolikkopeli-ammuskelu aiheuttaa pelinäkymässä esiintyvää jatkuvaa hahmojen liikettä sekä tarvetta tarkkailla pelaajan nopeitakin komentoja pelihahmolle. Tämänluonteinen pelilaji vaatii toimiakseen erillisen luupin, joka tarkkailee pelaajan komentoja sekä näkymässä esiintyvien olioiden tilaa (esimerkiksi ovatko viholliset vielä ”elossa”). Tämän sovelluksen toteutuksessa kyseinen luuppi toteutettiin omaan luokaansa, ja luuppia ajetaan moottorin toimesta pelinäkymän ollessa sovellusaktiviteetin näkyvänä osana.

Alla näkyvässä kuvassa näkyvät pelinäkymän näkyvät elementit:



KUVA 10. Kuvakaappaus esimerkkisovelluksen peli-näkymästä emuloidussa Android-laitteessa.

Pelinäkymä koostuu taustakuvansa lisäksi neljästä näkyvästä elementistä: pelihahmosta, ammuksista, vihollishahmoista sekä ohjaukseen käytettävistä painikkeista.

Pelihahmon ohjauspainikkeet toteutettiin luomalla pelinäkymään niin sanottu HUD (heads-up display), jonka toteuttamiseen AndEngine tarjosi valmiin HUD-olion. Yleensä HUD sisältää erilaisia näyttöjä ja lukemia, joiden tietojen välittäminen pelaajalle kesken pelikokemuksen on tärkeää: tietoina voidaan välittää esimerkiksi pelaajan pelissä ansaitsemat pisteet. Tämän työn esimerkkisovelluksessa HUD:iin sisällytettiin ainoastaan ohjaukseen liittyvät toiminnot, varsinaisen pelin tavoitteen jäädessä jatkokehityksessä toteutettavaksi. Pelinäkymässä tämä ”näkymän päällimmäiseksi kerrokseksi” asettuva ohjainnäkymä luodaan pelinäkymän rakentajassa kutsuttavalla funktiolla, jossa määritetään molemmat painikkeet, näiden paikat näytöllä ja asetetaan niille toiminnot. Painikkeita painettaessa kutsutaan pelaajan luokassa toteutettua liikkumisfunktiota.

Pelihahmon toimintaominaisuus, vihollisten tuhoaminen ampumalla, toteutettiin myös pelinäkymä-luokkaan. Pelaajan aseen käyttäminen aktivoidaan koskettamalla laitteen näyttöä (muualta kuin ohjauspainikkeiden kohdalta). Pelinäkymässä toteutettu näytön koskettamiseen reagoiva funktio tarkastaa näyttöä kosketettaessa, onko aseessa ”jäähdytys” voimassa: mikäli tämä ominaisuus on ”pois päältä”, kutsutaan pelaaja-luokasta varsinaista ampumisen toteutusta.

Pelihahmon omassa luokassa määritellään pelihahmon grafiikat sekä oletussijainti pelinäkymässä. Toteutetussa demoversiossa pelihahmon grafiikkana toimii AndEnginen valmiina tarjoama suorakulmio-kuvio. Pelihahmon ominaisuuksien lisäksi luokka sisältää toteutukset pelihahmon liikkumiselle (sisältäen esimerkiksi rajatarkastukset) sekä varsinaiselle ampumiselle. Hahmon oletussijainti pelin alussa, pelinäkymän alalaidassa keskellä, tarkastetaan ja korjataan tarvittaessa omalla funktiollaan.

Pelin vihollisten toteutus on toteutettu kahteen erilliseen luokkaan: varsinaisessa vihollis-luokassa määritellään pelihahmon grafiikat (pelihahmon tapaan suorakulmio-kuviona toteutettu) sekä energia (kuinka monta ammuksen osumaa yksittäinen vihollinen kestää). Lisäksi tämä luokka sisältää toteutukset vihollisolion aktivoimiselle ja mitätöinnille pelinäkymässä sekä pelihahmon ammuksen osuman tarkistukselle.

Useaa vihollisoliota varten sovellukseen luotiin oma muistivaranto omaan luokkaansa. Tämä säiliö toteutettiin AndEnginen omalla GenericPool-säiliöluokalla. Säiliölle määritetään siis tietty lukumäärä vihollisolioita, joita käytetään yhdellä pelikerralla. Jokaisen pelikerran oliot noudetaan tästä yksittäisestä säiliöstä.

Pelihahmon käyttämät ammukset on toteutettu kahteen erilliseen luokkaan: varsinaisessa ammusluokassa määritellään yksittäisen ammusolion grafiikat (tämäkin on toteutettu AndEnginen tarjoamana suorakulmio-kuviona).

Useaa ammusoliota varten sovellukseen luotiin oma muistivaranto omaan luokkaansa, samaan tapaan kuin usealla vihollisoliolle. Tämäkin säiliö toteutettiin AndEnginen omalla GenericPool-säiliöluokalla. Säiliölle määritetään tietty lukumäärä ammusolioita, joita käytetään yhdellä pelikerralla. Jokaisen pelikerran oliot noudetaan tästä yksittäisestä säiliöstä.

Aseen ”jäähdyttäminen” on toteutettu omaan luokkaansa. Aseen jäähdyttämisen (toisin sanoen aseen käytön rajoittamisen) toteuttamiselle on kaksi erillistä syytä: pelikokemuksen näkökulmasta aseen rajaton käyttömahdollisuus helpottaa pelaamista liian paljon: vaikkei pelin nykyisellään toteutetussa versiossa varsinaista vastusta olekaan, otettiin aseen jäähdyttäminen jatkokehittämisen kannalta jo tässä vaiheessa toteutuslistalle. Lisäksi toteutuksen taustalla on tekninen syy: muistinkäytön pienentäminen onnistuu osaltaan hyvin turhien olioiden kutsumisen rajoittamisella. Tämä saavutetaan osaltaan aseen jäähdyttämisenkin toteuttamisella, mutta myös edellä esitellyt ammukset ja viholliset varaavat muistisäiliöt helpottavat muistinkäyttöä.

6.2. Pelin jatkokehitykseen liittyviä ideoita

Aikataulun kireys aiheutti monen pelielementin yksinkertaistamisen tai kokonaan karsimisen: työn lopputuloksesta puuttuvat esimerkiksi kaikki varsinaiset grafiikat ja äänitehosteet. Myös esimerkiksi pelitason läpäisystä käynnistettävä pistenäkymä toteutettiin ilman varsinaista käyttötarkoitusta: olennainen elementti toteutettiin teknisistä syistä, mutta varsinaisessa sovelluksen julkaisuversiossa pistenäytössä tulisi esitellä esimerkiksi kentän läpäisynopeus, pelaajan osumatarkkuus tai jokin vastaava pelaamisarvoa nostava tavoite.

Edellä mainittujen grafiikoiden, äänien ja ”pelin tavoitteen” lisäksi jatkokehityksessä huomioon voitaisiin ottaa myös pelin jatkuvuus: useiden kenttien lisäksi jonkinlainen kokonaispistemäärä ja parhaat tulokset sisältävä tietokokonaisuus (sovelluksen sisäinen tietokanta tai vastaava) voisivat olla mahdollisia ”koukkuja” julkaisukelpoiseen pelisovellukseen.

Yksi jatkokehityksen kohde olisi sovelluksen jakaminen useaan aktiviteettiin: kun esimerkiksi pelinäkymä toimisi omassa aktiviteettissaan, olisi pelin pysäyttäminen taukovalikkoon siirtymistä varten mahdollista toteuttaa. Tällöin taukovalikon sulkeutuessa voitaisiin palata tauolle jätettyyn pelinäkymä -aktiviteettiin, ilman että keskeytetty pelikerta ”nollautuisi”. Tällä hetkellä pelikerta ”nollataan” muistiteknisistä syistä aina laitteen back-painiketta painettaessa: tämä on pelin esiaste -versiossa perusteltua mutta varsinaisessa valmiissa sovelluksessa ratkaisu ei olisi hyväksyttävä.

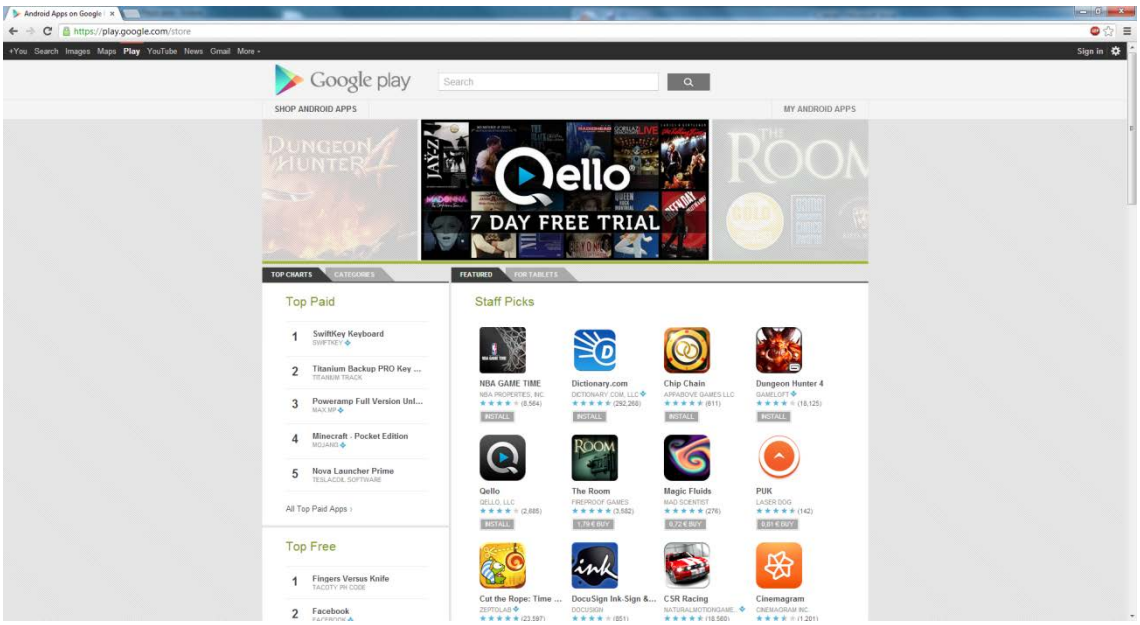
7 SISÄLTÖPALVELU GOOGLE PLAY JA VALMIIN SOVELLUKSEN JULKAISUN VAIHEET

Tässä luvussa esitellään Google Play -sisältöpalvelua, sen ominaisuuksia ja historiaa. Lisäksi käydään vaiheittain lävitse toimenpiteet, joita valmiin sovelluksen julkaisemiseksi palvelussa tarvitaan. Maksullisten sovellusten julkaisemiseen liityviä asioita katsotaan hyvin yleisellä tasolla asioita, keskittyen tärkeimpiin asioihin (varsinaisten sopimusehtojen ollessa erittäin laajoja ja yksityiskohtaisia).

Android-käyttöjärjestelmää käyttäviin laitteisiin on mahdollista asentaa ja niissä suorittaa myös kolmannen osapuolen jakelemia sovelluksia (toisin sanoen täysin Googleen liittymättömien tahojen julkaisuja). Kuitenkin, ensisijaisena sovellusten latauslähteenä voidaan suosia käyttöjärjestelmän ylläpitäjän omaa sisältöpalvelua (tässä tapauksessa siis Google Play), joka vakaan julkaisualustan lisäksi toimii erinomaisena lähtötason markkinointikanavana. Siksi Android-sovelluskehittäjän on hyvä selvittää julkaisun vaiheet Google Play -palvelussa, tulevaisuuden julkaisuja silmällä pitäen.

7.1. Google Play -sisältöpalvelu

Google Play on Googlen omistama digitaalinen sisältöpalvelu. Palvelu tarjoaa etupäässä Android-käyttöjärjestelmille suunnattuja sovelluksia. Niiden lisäksi palvelun sisältö tarjoaa myös esimerkiksi musiikkia, elokuvia sekä e-kirjoja. Palvelu syntyi maaliskuussa 2012, kun Googlen omistamat Android Market -sovelluspalvelu, musiikkia tarjonnut Google Music sekä e-kirjaliike eBookstore yhdistyivät. (Google Official Blog: Introducing Google Play: All your entertainment, anywhere you go)



KUVA 11. Google Play -sisältöpalvelun etusivunäkymä (Google Play)

Palvelun sisältö kattaa sekä ilmaisia että maksullisia sovelluksia. Ilmaiset sovellukset ovat usein mainosrahoitteisia: tällöin sovellusta käytettäessä mainostajien ilmoituksia saattaa näkyä esimerkiksi laitteen laidassa liukuvana mainosbannereina, joiden tuotto käytetään esimerkiksi sisältöpalvelun ylläpitämiseen ja siten mahdollistetaan sen käyttämisen maksuttomuus.

Google Play -kauppasovellus löytyy useimmista Android-mobiililaitteista valmiiksi esiasennettuna; tämä applikaatio mahdollistaa palvelun tuotteiden suoraan laitteeseen lataamisen ja asentamisen. Toinen mahdollisuus hankkia sisältöä Google Playsta on hankkia halutut tuotteet perinteisen tietokoneen avulla Google Play -web-sivustolta ja lataamalla ne tietokoneelle. Käyttövaatimuksena sisältöpalvelulle kohdelaitteesta riippumatta ovat internetyhteys sekä voimassaoleva Google-tili (koskien palvelun asiakkaita). (Google Play: Lataaminen)

7.2. Julkaisun vaiheet Google Play -palvelussa

Sovellusten julkaisemiseksi tulee sovelluskehittäjän toimia Googlen toiminta-alueella: Google Play -tukisivustolla on listattuna jakeluoikeudet omaavat maat, joissa sovellusten jakelijat ovat oikeutettuja toimintaansa (erillissä listauksissa maksulliset / ilmaissovellukset) sekä mahdolliset jakelukohdemaat (erillisissä listauksissa maksulliset/ilmaissovellukset). (Google Play: Supported locations for distributing applications)

Sovellusten julkaiseminen Google Play -palvelussa voidaan jakaa karkeasti kolmeen osaan: Google Play Developer Console -tilin luomiseen ja rekisteröimiseen, Google Checkout Merchant Account -tilin luomiseen (tämä ainoastaan, mikäli sovelluksesta julkaistaan maksullinen versio) sekä Google Play Developer Console -konsolin työkaluihin tutustumiseen ja varsinaisen julkaisun suorittamiseen.

Jokainen palvelussa julkaiseva taho tarvitsee toimiakseen Google Play Developer Console -tilin, jonka avulla voi rekisteröityä myös julkaisijaksi. Tämä tapahtuu itse Google Play Developer Console -konsolin kautta. Tässä vaiheessa julkaisijaksi haluava syöttää palveluun omat perustietonsa, kuten nimensä ja sähköpostiosoitteensa. Kaikki perustiedot on muokattavissa vielä jälkikäteenkin (tämä mahdollistaa esimerkiksi muuttuneiden tietojen muokkamisen).

Syötettyään tietonsa hakemuksen täyttäjän tulee hyväksyä Googlen oma Kehittäjän jakelusopimus. Kyseessä on maakohtainen sopimus, joka oikeuttaa sekä velvoittaa sovelluksen kehittäjän Googlen toimivallan alaiseen toimintaan: sopimus esimerkiksi antaa kehittäjälle oikeuden julkaista sovelluksiaan palvelussa, mutta samalla valtuuttaa Googlen vastaamaan asiakkaiden hyvitysvaatimuksiin kehittäjän puolesta (sopimuksen edellyttämissä puitteissa). Julkaistavien sovellusten tulee jakelusopimuksen lisäksi noudattaa Google Playn Kehittäjen ohjelmasääntöjä sekä koti- että ulkomaisia vientilakeja. (Google Play: Kehittäjien jakelusopimus)

Sopimuksen hyväksymisen jälkeen hakijan tulee suorittaa 25 USD:n kertaluontoinen rekisteröintimaksu Google Checkout -palvelun avulla. Palvelu vaatii myös omat tunnuksensa. Tämän maksun lisäksi muita kuluja Google Playssä julkaisemisesta ei synny (poislukien esimerkiksi markkinointi ja vastaavat toimet). Google lupaa, että tilihakemuksen käsittelyprosessi kestää korkeintaan 48 tuntia. (Android Developer: Get Started with Publishing, Google Play: Developer Registration)

Mikäli sovelluksen aikoo julkaista maksuttomana, voi sen tehdä välittömästi Google Play Developer Console -tilin aktivoiduttua. Tämä tapahtuu Developer Console -konsolin Upload Applications -kohdan kautta. Julkaisuun tarvitaan itse julkaistava sovellus yhtenä .apk-muotoisena pakettina, jonka koko on korkeintaan 50 Mt, (vaihtoehtoinen) sovellusluonnoksen .apk-tiedosto, vähintään kaksi kuvakaappausta itse

sovelluksesta käytössä sekä korkearesoluutioinen sovellusikoni. Näiden vähimmäisvaatimusten lisäksi sovelluksen oheismateriaalina on mahdollista julkaista myynninedistysmielessä esimerkiksi grafiikkaa (mainosjulisteet, lisäkuvakaappaukset jne.) sekä promootiovideoita. (Google Play: Sovellusten lähettäminen)

Lisäksi jaettavasta sovelluksesta tulee ilmoittaa vähintään perustiedot: sovelluksen käyttökieli, nimi sekä kuvaus. Tarvittaessa tietoihin tulee täydentää ilmoitukset viimeaikaisista muutoksista sovelluksen uusimmassa versiossa, sekä halutessaan vapaaehtoinen mainosteksti. Sovelluksen tyypiksi valitaan kahdesta vaihtoehdosta sopiva: sovellus tai peli. Lisäksi sovellukselle tulee valita sopiva luokka valmiiksi asetetuista vaihtoehdoista. (Google Play: Sovellusten lähettäminen)

Google Playsta ladattavat sovellukset ovat pakattu .apk-päätteiseen tiedostoon, jonka laitteen ohjelmisto purkaa latauksen jälkeen ja asentaa sovelluksen laitteeseen. Esimerkiksi tässä työssä käytetty Eclipse-kehitysympäristö kääntää sovellukset suoraan apk-paketiksi: täten valmiiksi koettu ja ohjelmakoodin kääntäjässä onnistuneesti kääntynyt sovellus voidaan siirtää suoraan sisältöpalveluun julkaistavaksi.

7.3. Maksullisten julkaisujen julkaisemiseen liittyvät lisätoimenpiteet Google Play -palvelussa

Mikäli sovelluksensa haluaa julkaista maksullisena, tulee julkaisijalla olla Google Play Developer Console -tilin lisäksi Google Checkout -kauppiastili. Sen rekisteröiminen onnistuu oman Google Play -tilin Muokkaa profiilia -osion kautta.

Rahan astuessa kuvioihin myös toimintamallit monimutkaistuvat huomattavasti: esimerkiksi maksullisten sovellusten jakelu on huomattavasti rajatumpaa kuin ilmaisten sovellusten tapauksessa. (Google Play: Supported locations for merchants)

Googlen osuus myytyjen tuotteiden tuotosta on 30% tuotteen nettohinnasta. Tällä osuudella katetaan sovellusten myymisestä syntyvät tapahtumakulut. Kehittäjälle maksetaan siis 70% myytyjen tuotteiden nettohinnoista sekä perittävät verot (Suomen laissa ALV 24% käyttötuotteissa (tarkasta), tulee sisällyttää kauppahintaan). Kehittäjän tulee sisällyttää kaikki verot (myös ALV) tuotteen hintaan silloin, kun sitä hinnoittelee. Google maksaa tuotteen julkaisijalle tämän ansaitseman saldon kuukausittain

(normaalina arkipäivänä, mikäli saldo maksuajankohtana on vähintään 1 USD). (Google Play: Transaction Fees, Google Play: Payouts FAQ, Google Play: Specifying tax rates)

Ehkä tärkein yksittäinen huomio Google Checkout -kauppiastilin yhdistämisestä omaan Google Play Developer Console -julkaisutiliin on sen kertaluonteisuus sekä toimenpiteen peruuttamattomuus: kerran toisiinsa yhdistetyt tilit tulevat aina olemaan riippuvaisia toisistaan, eikä niitä voida poistaa tai muuttaa. Ainoa mahdollisuus uuden tilin käyttämiseen on luoda sellainen alusta alkaen (tämä toimenpide pitää edelleen sisällään myös 25 USD:n rekisteröitymismaksun). Toinen maininnan arvoinen seikka on se, ettei kerran ilmaisena julkaistua sovellusta voi muuttaa enää maksulliseksi. Ainoa mahdollisuus on julkaista kehitetty .apk-paketti uudella nimellä, maksullisena versiona. (Google Play: Create a Google Checkout account from the developer console, Android Developer: Publishing Checklist for Google Play)

8 POHDINTA

Opinnäytetyö työtehtävänä mahdollisti erinomaisesti tutustumisen pelimoottorien maailmaan. Sen lisäksi, että peliteollisuus on mahdollinen tulevaisuuden työllistäjä omalla kohdallanikin, on pelikehitys muutenkin ohjelmistotekniikan opiskelijalle oiva tapa tutustua ohjelmistojen tekemiseen: esimerkiksi tässä työssä ymmärrykseni myös yleisesti valmiita ohjelmistokehyksiä kohtaan kasvoi merkittävästi.

Lisäksi oman aiheen laajentaminen mahdollisti tutustumisen Google Playn julkaisun vaiheisiin, niin ilmaisten kuin maksullistenkin sovellusten osalta. Työtä tehdessä selvisi, etteivät asiat ole niin yksinkertaisia kuin voisi kuvitella, eikä esimerkiksi varsinainen ”rahan tahkoaminen” mobiilipeleillä ole mitenkään yksiselitteistä. Ymmärrys tuotteen julkaisua ja talousnäkökulmaakin kohtaan on kuitenkin myös osa insinöörin ammatillista näkökenttää ja siksi koin tärkeäksi perehtyä myös tähän osioon hieman tarkemmin.

Pelimoottori osoittautui työskentelyalustana jokseenkin haasteelliseksi. Haasteellisuuteen vaikutti varmasti oma alhainen kokemustasoni etenkin peliohjelmoinnissa. Roolinsa oli kuitenkin myös AndEnginen kommentoinnin ja dokumentoinnin täydellisellä puuttumisella. Vaikka esimerkkisovellus on tämän työn kirjoitushetkellä hyvin vaatimaton, ilman tukea pelimoottorin viralliselta keskustelupalstalta sovellus ei välttämättä olisi vieläkään pelaamiskelpoinen. Tavoite täysin valmiista omasta sovelluksesta jäi toteutumatta: tämä oli kuitenkin itselleni asettama ”osatavoite” ja varsinainen päätavoite, tutustua Android-pelikehityksen työkaluihin ja -menetelmiin, saatiin toteutumaan mielestäni hyvin. AndEnginen soveltuvuutta pelikehitykseen en voi kyseenalaistaa, varsinkaan kokeneen ohjelmoijan käsissä. Kuitenkin, etenkin aloittelijalle (jollaiseksi myös itseni luen peliohjelmoijana), AndEngine soveltunee todelliseksi työkaluksi vain mikäli ohjelmoijalla on aikaa ja palavaa intoa pelikehitystä kohtaan.

Suurin rajoittava tekijä työni teossa oli aika. Ymmärrys ajan kuluvuutta ja ajan tehokkaan käytön tärkeyttä kohtaan kasvoi valtavasti. Pienenkin sovelluksen suunnittelulle on varattava aikaa ja varsinaisen sovelluksen tekemisen lisäksi aikaa kuluu valtavasti esimerkiksi työkalujen ja menetelmien valitsemiseen. Pienikin peliprojekti on aikaa vievä, eikä jokaisen projektin osan suunnittelua voi ajallisesti

ennakoida kovinkaan tarkasti ennen työvaiheen aloittamista (varsinkaan aloittelijana). Itse yllätyin ”kirjallisen” osuuden työläydestä, enkä osannut valmistautua ajan kulumisen jakautumiseen lähes tasan ohjelmoinnin ja teoriaan tutustumisen sekä kirjoittamisen välillä.

Tekniikoihin ja välineisiin tutustumisen lisäksi tämä työ opetti valtavasti omista taidoista ohjelmistoalan tulevana ammattilaisena. Ymmärrys taitojani kohtaan kasvoi ja ennen kaikkea monesti toisteltu viisaus todettiin paikkaansa pitäväksi: parhaiten oppii tekemällä. Vaikka oma osaamiseni on vasta kehittymässä, on jo nykyisillä taidoillani mahdollista saada aikaan toimiva sovellus ja matkan varrella kertyneillä tiedoilla mahdollisuus kehittyä edelleen.

Kehitystyötä jäi pelisovelluksen suhteen edelleen runsaasti: pelin grafiikat ja äänet eivät todennäköisesti synny ohjelmoijan käsissä helposti, mutta toivottavasti nekin toteutuvat jossain vaiheessa. Myös ominaisuuksien lisääminen sovellukseen on tarpeen ennen varsinaista julkaisua: tätä pohdintaa kirjoitettaessa ideoita jatkokehitykselle on syntynyt ja osa on jo toteutuksen alaisena.

Kaiken kaikkiaan opinnäytetyölle asettamani tavoitteet saavutettiin kaikesta huolimatta mielestäni hyvin. Vaikka työssä toteutettu sovellus jäi tulevaisuuden julkaisuversioon verrattaessa vaatimattomaksi, oman tekemisen tuloksen näkeminen kasvatti itsevarmuutta ja innostaa varmasti jatkamaan myös tämän sovelluksen kehittämisen parissa. Joka tapauksessa tämän työn kautta karttuneet tietoni ja taitoni auttavat minua varmasti tulevaisuuden työtehtävissäni, oli vastuullani sitten pelisovelluksen tai minkä tahansa muun ohjelmistoprojektin tekeminen.

LÄHTEET

AndEngine: Forum. Luettu 2.5.2013.
http://www.andengine.org/forums/

Android Developer: Activities. Luettu 3.5.2013
http://developer.android.com/guide/components/activities.html

Android Developer: App Framework. Luettu 2.5.2013.
http://developer.android.com/about/versions/index.html

Android Developer: AVD Manager. Luettu 3.5.2013.
http://developer.android.com/tools/help/avd-manager.html

Android Developer: Dashboards. Luettu 15.4.2013.
http://developer.android.com/about/dashboards/index.html

Android Developer: Developer Tools. Luettu 2.5.2013.
http://developer.android.com/tools/index.html

Android Developer: Get Started with Publishing. Luettu 3.5.2013.
http://developer.android.com/distribute/googleplay/publish/register.html

Android Developer: Get the Android SDK. Luettu 5.5.2013.
http://developer.android.com/sdk/index.html

Android Developer: Managing Virtual Devices. Luettu 3.5.2013.
http://developer.android.com/tools/devices/index.html

Android Developer: Publishing Checklist for Google Play. Luettu 3.5.2013.
http://developer.android.com/distribute/googleplay/publish/preparing.html

Android Developer: System Architecture. Luettu 3.5.2013
http://developer.android.com/images/system-architecture.jpg

Android Developer: Tools Help. Luettu 4.5.2013.
http://developer.android.com/tools/help/index.html

Android Developer: Using Hardware Devices. Luettu 4.5.2013.
http://developer.android.com/tools/device.html

Android Developers Blog: Dalvik JIT. Luettu 3.5.2013.
http://android-developers.blogspot.fi/2010/05/dalvik-jit.html

Android Open Source Project. Luettu 3.5.2013.
http://source.android.com

Android Open Source Project: About the Android Open Source Project. Luettu 5.5.2013. http://source.android.com/about/index.html

Android Open Source Project: Compatibility Program Overview. Luettu 3.5.2013.
http://source.android.com/compatibility/overview.html

Android Open Source Project: Philosophy and Goals. Luettu 15.4.2013.
http://source.android.com/about/philosophy.html

Business Week: Google Buys Android for Its Mobile Arsenal. Luettu 3.5.2013.
http://www.businessweek.com/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal

Eclipse Project: About the Eclipse Foundation. Luettu 3.5.2013.
http://www.eclipse.org/org/

Eclipse Project: Downloads. Luettu 3.5.2013.
http://www.eclipse.org/downloads/

Ehringer David: The Dalvik Virtual Machine Architecture. Luettu 6.5.2013.
http://show.docjava.com/posterous/file/2012/12/10222640-The_Dalvik_Virtual_Machine.pdf

eLinux: Android Dalvik VM. Luettu 2.5.2013.
http://elinux.org/Android_Dalvik_VM

Gamasutra: The Designer's Notebook: Sorting Out the Genre Muddle. Luettu 5.5.2013.
http://www.gamasutra.com/view/feature/132463/the_designers_notebook_sorting_.php

Game Career Guide: What is a Game Engine?. Luettu 4.5.2013.
http://www.gamecareerguide.com/features/529/what_is_a_game_.php

Github: Andengine / RatioResolutionPolicy.java. Luettu 3.5.2013.
https://github.com/nicolasgramlich/AndEngine/blob/GLES2/src/org/andengine/engine/options/resolutionpolicy/RatioResolutionPolicy.java

GitHub: AndEngine / Readme. Luettu 4.5.2013.
https://github.com/nicolasgramlich/AndEngine#readme

GitHub: Nicolas Gramlich. Luettu 6.5.2013.
https://github.com/nicolasgramlich/

Google Official Blog: Introducing Google Play: All your entertainment, anywhere you go. Luettu 3.5.2013.
http://googleblog.blogspot.fi/2012/03/introducing-google-play-all-your.html

Google Play. Luettu 15.4.2013.
https://play.google.com/store

Google Play: Create a Google Checkout account from the developer console. Luettu 2.5.2013.
https://support.google.com/googleplay/android-developer/answer/2972701?hl=en

Google Play: Developer Registration. Luettu 5.5.2013.
https://support.google.com/googleplay/android-developer/answer/113468?hl=en

Google Play: Kehittäjien jakelusopimus. Luettu 3.5.2013.
http://play.google.com/about/developer-distribution-agreement.html

Google Play: Lataaminen. Luettu 3.5.2013.
http://support.google.com/googleplay/bin/answer.py?hl=fi&answer=113409

Google Play: Payouts FAQ. Luettu 3.5.2013.
https://support.google.com/googleplay/android-developer/answer/173779?hl=en&ref_topic=15867

Google Play: Sovellusten lähettäminen. Luettu 5.5.2013.
https://support.google.com/googleplay/android-developer/answer/113469?hl=fi&ref_topic=2365624

Google Play: Specifying tax rates. Luettu 5.5.2013.
https://support.google.com/googleplay/android-developer/answer/138000?hl=en&ref_topic=15867

Google Play: Supported locations for distributing applications. Luettu 6.5.2013.
http://support.google.com/googleplay/android-developer/answer/138294?hl=en&ref_topic=2365624

Google Play: Supported locations for merchants. Luettu 3.5.2013.
http://support.google.com/googleplay/android-developer/answer/150324?hl=en&ref_topic=15867

Google Play: Transaction Fees. Luettu 3.5.2013.
https://support.google.com/googleplay/android-developer/answer/112622?hl=en&ref_topic=15867

IBM developerWorks: New to Java programming. Luettu 5.5.2013.
http://www.ibm.com/developerworks/java/newto/?ca=jv_digdeep

IBM developerWorks: Java theory and practice: A brief history of garbage collection. Luettu 5.5.2013.
http://www.ibm.com/developerworks/java/library/j-jtp10283/index.html

Khronos: OpenGL ES. Luettu 5.5.2013.
http://www.khronos.org/opengles/

Open Handset Alliance: Industry Leaders Announce Open Platform for Mobile devices. Luettu 3.5.2013. http://www.openhandsetalliance.com/press_110507.html

Open Handset Alliance: Open Handset Alliance Releases Android SDK. Luettu 3.5.2013. http://www.openhandsetalliance.com/press_111207.html

Open Handset Alliance: Overview. Luettu 3.5.2013.
http://www.openhandsetalliance.com/oha_overview.html

Oracle Technology Network: Java. Luettu 4.5.2013.
http://www.oracle.com/technetwork/java/index.html

Oracle Technology Network: The History of Java Technology. Luettu 5.5.2013.
http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html

The Telegraph: Samsung to launch Android-powered fridge. Luettu 3.5.2013.
http://www.telegraph.co.uk/technology/samsung/9850412/Samsung-to-launch-Android-powered-fridge.html

Vogella.com: Eclipse IDE Tutorial. Luettu 3.5.2013.
http://www.vogella.com/articles/Eclipse/article.html

W3C: Standards. Luettu 4.5.2013.
http://www.w3.org/standards/

W3C: XML Essentials. Luettu 4.5.2013.
http://www.w3.org/standards/xml/core

Web-Dot-Dev: Java Advantages and Disadvantages. Luettu 5.5.2013.
http://www.webdotdev.com/nvd/content/view/1042/204/]