PELIOHJELMOINTI ANDROIDILLE ANDENGINEÄ KÄYTTÄEN

Janne Hakala

Opinnäytetyö
Marraskuu 2013
Tietojenkäsittely
Ohjelmistotuotanto

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Ohjelmistotuotanto

HAKALA, JANNE:
Peliohjelmointi Androidille AndEngineä käyttäen

Opinnäytetyö 32 sivua
Marraskuu 2013

Opinnäytetyön tavoitteena oli tutkia miten AndEngine-pelimoottoria voidaan käyttää hyväksi Android-peliohjelmoinnissa. Tarkoituksena oli tutustua AndEnginen tarjoamiin peliohjelmointia helpottaviin ominaisuuksiin ja luoda näiden tutkimusten pohjalta yksinkertainen esimerkkipeli, jossa käytetään näitä ominaisuuksia. Opinnäytetyöllä ei ollut toimeksiantajaa.

Opinnäytetyön ensimmäisissä kappaleissa käydään läpi mobiilipelien historiaa ja nykytilannetta sekä tutustutaan yleisesti Androidiin ja sen peliohjelmointiin. Myöhemmin esitellään AndEngine ja sen tarjoamia ominaisuuksia. Käytännön osuudessa luotiin pieni peli, jossa näitä AndEnginen ominaisuuksia käytettiin. Esimerkkipelin luonnin jälkeen tarkasteltiin pelin toimivuutta erilaisilla Android-laitteilla.

Peliä tehdessä selvisi, että AndEngine helpottaa useita käytännön asioita mobiilipelikehityksessä. Testausvaiheessa selvisi, ettei AndEnginellä tehty peli toiminut ollenkaan vanhimmalla testauslaitteella. Uudemmat laitteet sen sijaan pyörittivät peliä hyvin, vain eri resoluutioille skaalautuminen aiheutti pieniä graafisia ongelmia.

Asiasanat: Android, Andengine, peliohjelmointi, Java

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree programme in Business Information Systems
Software Development

HAKALA JANNE:
Android game development using AndEngine

Bachelor's thesis 32 pages
November 2013

The purpose of this thesis was to research how the AndEngine game engine can be used in Android game development. The objective of this thesis was to examine the features AndEngine provides for game development and make an example game using these features. This thesis did not have a client.

The first chapters of this thesis walks through the history and current situation of mobile games and also provides a general view of Android and game development on the platform. AndEngine and its features are introduced later in the thesis. The practical part of the thesis consists of making a small example game using AndEngine's features. After the creation of the example game, the game was tested on various Android devices.

While programming the game, it became clear that AndEngine makes Android game development easier. While testing the example game on different devices it was concluded that the game did not work on the oldest device. Newer devices ran the game really well, only scaling the game to different resolutions resulted in some graphical issues.

Key words: Android, Andengine. game development, Java

SISÄLLYS

LYHENTEET JA TERMIT

SDK	Software development kit.
NDK	Native development kit.
Roottaus	Android-laiteen pääkäyttäjäoikeuksien hankkiminen.
Plugin	Lisäosa.
Git	Git-versionhallintajärjestelmä.
Metodi	Yksittäinen aliohjelma, jota kutsutaan muista osista ohjelmaa.

1 JOHDANTO

Älypuhelimet ja tabletit kasvattavat nykyään markkinaosuuttaan yhä laajenevissa määrin. Yhä useampi ihminen ostaa älypuhelimia ja vanhempien perusmallien myynti laskee, myös tablettien myynti on kovassa kasvussa. Tästä syystä myös peliteollisuus mobiililaitteille kasvaa. Suomestakin on tullut kaksi todella vahvaa yritystä mobiilipelimarkkinoille; Rovio ja Supercell. Mobiilipelimarkkinoille pääseminen ja menestyminen vaatii idean lisäksi myös ohjelmointiosaamista ja tähän helpotusta tuo AndEngine.

Tämän opinnäytetyön tavoitteena on tutkia miten AndEngine-pelimoottoria voidaan käyttää hyväksi Android-peliohjelmoinnissa. Tarkoituksena on tutustua AndEnginen tarjoamiin peliohjelmointia helpottaviin ominaisuuksiin ja luoda näiden pohjalta yksinkertainen esimerkkipeli. Opinnäytetyöllä ei ole toimeksiantajaa.

Aluksi käydään läpi hieman mobiilipelien yleistä historiaa ja nykytilannetta, tämän jälkeen esitellään alusta, jolle pelimoottori on suunnattu. AndEngine tarjoaa huomattavan määrän ominaisuuksia, jotka helpottavat yleisiä peleissä käytettäviä asioita. Näiden kirjoittaminen alusta asti käsin on työlästä ja aikaa vievää. Näihin ominaisuuksiin tutustutaan alustan esittelyn jälkeen.

Android-laitteita on todella paljon erilaisia ja niissä on huomattavan isoja eroja suorituskyvyssä. Esimerkkipelissä käytetään erilaisia AndEnginen ominaisuuksia ja peliä testataan erilaisilla Android-laitteilla. Valmiin pelin testauksessa selviää, kuinka hyvin AndEngine skaalautuu eri resoluutioille ja kuinka paljon eri lailla peli käyttäytyy erilaisissa laitteissa.

2 YLEISTÄ MOBIILIPELEISTÄ

2.1 Mobiilipelien historiaa

Mobiilipeli terminä tarkoittaa liikkuvaa peliä, oli se sitten käsikonsoli tai matkapuhelimella pelattava peli. Tässä opinnäytetyössä mobiilipelillä tarkoitetaan matkapuhelimella (älypuhelimella) pelattavia pelejä.

Mobiilipelit saivat käytännössä alkunsa vuonna 1997, jolloin Nokia 6110 puhelimessa oli yksinkertainen matopeli. Siinä ohjattiin matoa numeronäppäimillä ja pelin tarkoituksena oli syödä ruudulle ilmestyviä omenoita, samalla seiniä ja itse matoa vältellen. Pelissä oli erilaisia vaikeustasoja ja pisteytys vaikeustason mukaan. Aina kun mato söi omenan, mato kasvoi ja täten madon liikkuma-alue pieneni. Tästä johtuen pisteitä ei voinut saada tiettyä määrää enempää, sillä mato täytti koko pelialueen ja peli loppui. Pelin grafiikka oli todella yksinkertaista, mato koostui mustista yhdessä olevista neliöistä ja omenat olivat neljän pikselin muodostama salmiakkikuvio. Vaikka peli oli hyvin yksinkertainen, se oli aikanaan todella suosittu mobiilipeli.

Myöhemmin matkapuhelimet kehittyivät tehoiltaan ja näyttöihin tuli värejä. Tämä kehitys innoitti pelikehittäjiä luomaan peleihin parempia grafiikoita ja muita ominaisuuksia.

Javan tulo mobiilimarkkinoille toi aivan uusia ulottuvuuksia mobiilipelien kehityksessä. Yhtäkkiä kuka tahansa pystyi tekemään itse mobiilipelejä, tarvitsi vain opetella käyttämään JavaME:tä. JavaME-tyyppiset pelit ja sovellukset toimivat merkkiriippumattomasti puhelimissa, joissa Java-tuki oli. Ongelmana yksityishenkilöiden tekemissä peleissä ja sovelluksissa oli jakelu. Tähän ongelmaan ratkaisun toi palvelu nimeltä GetJar. GetJar-palveluun pystyi lähettämään omia pelejä ja niiden lataaminen oli tehty helpoksi puhelimen oman selaimen kautta.

2.2 Mobiilipelien nykytilanne

Tällä hetkellä mobiilisovellusten ja -pelien jakelu hoituu yleensä käyttöjärjestelmävalmistajan oman keskitetyn jakelujärjestelmän kautta. Applella on Appstore, Googlella on Google Play ja Nokialla on Ovi Store. Näistä palveluista on puhelimessa oma sovellus, jonka avulla voi ladata helposti uusia sovelluksia ja pelejä puhelimeen. Jakelujärjestelmissä on myös mahdollisuus ostaa pelejä ja sovelluksia esimerkiksi luottokorttimaksulla. Järjestelmiin voi kuka tahansa lähettää omia tuotoksiaan, kun maksaa aloitusmaksun. Nokian Ovi Storessa aloitusmaksu on yksi euro. (Nokia 2013.) Googlen Play palvelussa aloitusmaksu on $25. (Google 2013.) Apple AppStoressa on vuosimaksu, joka on $99 vuodessa. (Apple 2013.)

Google Play -palvelussa on mahdollista tehdä pelin sisäisiä maksuja, kuten esimerkiksi ostaa lisää pelimerkkejä peliin, jossa pelimerkkejä tarvitsee johonkin. Tämäntyylinen maksujärjestelmä mahdollistaa itse pelin ilmaisuuden, mutta pelissä ei pääse nopeasti tai ollenkaan eteenpäin ilman, että ostaa lisää pelimerkkejä. Monet pelikehittäjät julkaisevat niin sanotun Lite-version, jolla käyttäjä pääsee kokeilemaan rajoitettua versiota pelistä ennen ostoa. Tämä on omasta mielestäni jokseenkin kyseenalainen tapa; aiheutetaan mahdollinen riippuvuus peliin, sitten pyydetäänkin rahaa. Toisaalta tällä tavalla ei tarvitse maksaa pelistä turhaan, jos peli onkin käyttäjän mielestä tylsä, huono tai ei toimi käyttäjän laitteella.

Tämänhetkiset pelit ovat hurjasti edellä aikaisempia esimerkiksi JavaME:llä tehtyjä pelejä. Monista laitteista löytyy 3D-kiihdytyksellä varustettu näytönohjain, joka mahdollistaa hienojen 3D-pelien tekemisen. Tällä hetkellä on jopa tehty 3D MMORPG-tyylisiä pelejä. Näissä peleissä ohjataan hahmoa maailmassa, johon yhdistetään internetin kautta. Hahmot voivat muun muassa keskustella toisilleen ja tehdä tehtäviä yhdessä.

Joitakin PC:llä tai konsolilla pelattavia pelejä on myös käännetty puhelimille, kuten Grand Theft Auto: Vice City, Max Payne, Dead Space ja Need for Speed: Most Wanted. (Google Play 2013.)

3D-pelien lisäksi perinteiset 2D-pelit ovat vielä olemassa, suosion kriteereinä vaikuttavat olevan hahmojen söpöys, pelattavuus tai idea. Esimerkiksi Angry Birds on huippusuosittu peli, vaikka idea onkin erittäin yksinkertainen. Kirjoitushetkellä Angry Birds

Spacen ilmaisversiota on ladattu Google play palvelusta yli 50 miljoonaa kertaa. (Google Play 2013.)

3D- ja 2D-pelien suosiolla ei latausmäärien mukaan vaikuta olevan juurikaan eroja. Tästä voi päätellä, että idea ja pelattavuus vaikuttavat enemmän suosioon kuin pelin ulotteisuus.

3 ANDROID

3.1 Yleistä Androidista

Android on Googlen johtaman Open Handset Alliancen kehittämä mobiilikäyttöjärjestelmä. Androidin lähdekoodi on avoin ja se on julkaistu Apache-lisenssillä. Androidissa on Linux-ydin, jonka päällä on erilaisia kirjastoja sekä Dalvik virtuaalikone. Dalvik ajaa Java-ohjelmointikielellä kirjoitettuja ohjelmia. (Hildenbrand 2012.) Android-puhelimissa on yleensä valmistajan oma käyttöliittymä, joka voidaan korvata Google Play -palvelusta saatavilla launchereilla. Androidin rajoituksia voi myös poistaa roottaamalla, tämä avaa uusia ominaisuuksia kuten teemojen vaihto ja mainoksien poisto. Roottaaminen ei ole suositeltavaa jos ei löydy tarvittavaa osaamista, sillä väärin tehty roottaus saattaa aiheuttaa toimimattoman puhelimen.

International Data Corporationin (IDC) mukaan Android oli selvästi suosituin mobiilikäyttöjärjestelmä vuoden 2013 kolmannella neljänneksellä 81 prosentin markkinaosuudella (taulukko 1).

TAULUKKO 1. Markkinaosuudet 2013 kolmannella neljänneksellä (IDC 2013).

Käyttöjärjestelmä	Määrä (miljoonia)	Osuus (%)
Android	211,6	81,0
iOS	33,8	12,9
Window Phone	9,5	3,6
Blackberry	4,5	1,7
Muut	1,7	0,6
Yhteensä	149 041,8	100

Ensimmäinen Android-puhelin oli HTC Dream, Yhdysvalloissa nimellä T-Mobile G1. Se julkaistiin 23.8.2008. (HTC 2008.) Dreamissa oli Androidin versio 1.0 ja siihen tuli myöhemmin päivitys 1.1 versioon, joka oli ainoa Androidin päivitysversio, jonka nimi ei ollut jälkiruoka. Androidin versioiden nimet ovat nykyään makeita syötäviä versiosta 1.5 (Cupcake) lähtien. (Rubio 2011.)

Version 2.0/2.1 (Eclair) julkaisun jälkeen Androidia käyttäviä puhelimia alkoi ilmestyä kasvavalla tahdilla. Eclair-päivitys toi Androidiin uusia ominaisuuksia kuten Live Wallpapers (liikkuvat taustakuvat), navigoinnin joihinkin maihin ja HTML5 tuen. (Rubio 2011.)

Versio 2.2 (FroYo - Frozen Yoghurt) toi Androidiin USB- sekä WLAN yhteydenjakomahdollisuudet, näitä ominaisuuksia varten joutui ennen käyttämään erillisiä sovelluksia. FroYon mukana tuli myös Adobe Flash-tuki selaimeen ja automaattiset päivitykset Android Marketista ladatuille ohjelmille. (Rubio 2011.)

Versio 2.3 (Gingerbread) toi tuen suuremmille resoluutioille ja etukameroille. Päivitys sisälsi myös käyttöliittymän parannuksia. (Rubio 2011.)

Versio 3.0 (Honeycomb) oli suunnattu tablet-tyylisille Android-laitteille. Ulkoasu oli täysin erilainen aikaisempiin versioihin verrattuna. Androidin sisäänrakennettuja ohjelmia parannettiin isommille näytöille ja selain sai välilehdet ikkunoiden sijaan. (Rubio 2011.)

Versio 4.0 (Ice Cream Sandwich) oli suuri muutos 2.3 versioon verrattuna. Päivitys toi tullessaan näytönohjainkiihdytyksen normaaliin käyttöliittymään, joka nopeutti huomattavasti käyttöliittymää tavallisessa käytössä. ICS:n mukana tulivat myös virtuaaliset kosketusnäppäimet, joka mahdollisti laitteet, joissa ei ole fyysisiä menu-, back- ja home -nappeja. Päivityksessä oli myös paljon ulkonäköpäivityksiä sekä face unlock, jolla pystyy avaamaan laitteen lukituksen kuvaamalla kasvoja etukameralla. (The Verge 2013.)

Versio 4.1 (Jelly Bean) sisälsi Project Butter nimisen parannuksen, se nopeutti entisestään nopeutunutta käyttöliittymää. Päivityksessä tuli myös Google now, jonka tarkoituksena on luoda yhteenveto käyttäjälle tärkeistä asioista. Parannuksia tuli myös ulkonäköön, widgetteihin ja tekstinsyöttöön. Androidin Jelly Beanista on tehty useampi erinumeroinen versio: 4.1, 4.2 ja 4.3. Versiossa 4.2 yksi suurimpia lisäyksiä oli Miracast-tuki, jolla voi langattomasti lähettää videota tai ääntä suoraan laitteesta televisioon tai muuhun näyttöön. Muita uusia ominaisuuksia päivityksessä oli mm. käytettävyysparannukset ja usean käyttäjäprofiilin tuki. Version 4.3 pääasialliset muutokset olivat parempi useamman profiilin tuki ja OpenGL ES 3.0 -tuki. (The Verge 2013.)

3.2 Android-laitteet

Android-laitteiksi kutsutaan laitteita, joissa on käyttöjärjestelmänä Googlen Android. Yleisimmät Android-laitteet ovat älypuhelimia tai tabletteja. Android-laitteita aktivoidaan huimat 850 000 kappaletta päivittäin. Google Play palvelussa (ennen Android Market) on jo yli 450 000 sovellusta, joten kilpailu on hurjaa. Mobile World Congress 2012 tapahtumassa Android-ständillä oli yli 100 erilaista Android-laitetta nähtävillä. (Google Mobile Blog 2012.)

Androidia käyttäviä laitteita on listattuna Androidin omilla sivuilla 289 kappaletta. (Google 2013.) Tässä listauksessa ei ole mukana vanhempia laitteita, jotka eivät ole enää myynnissä, eikä listauksessa ole lähellekään kaikkia Android-laitteita, sillä listasta tulisi muuten käsittämättömän pitkä. Todellista laitteiden määrää ei voi arvioida, sillä niitä tullee tämänkin opinnäytetyön kirjoituksen aikana kymmeniä, ellei jopa satoja.

Älypuhelinten ja tablettien lisäksi Android on käytössä myös erikoisemmissa laitteissa, kuten Sonyn valmistamassa älykellossa, Samsungin digikamera/puhelinyhdistelmässä ja Parrotin ASTEROID autostereossa.

Suurimpien valmistajien laitteita kopioidaan lähes identtisellä ulkonäöllä ja jopa mallin nimellä. Nämä kopiot eivät kuitenkaan vaikuta olevan laitteistoltaan edes lähellä alkuperäisiä tuotteita. Esimerkkinä alkuperäinen on Samsung Galaxy Note 2, jossa on 5.5” näyttö, 1,6GHz neliydinprosessori ja 2 gigatavua RAM-muistia. (Samsung 2012.) Kopioversio on nimeltään "HDC Galaxy Note 2", jossa on pienempi näyttö, heikkotehoisempi prosessori ja vain 512Mt RAM-muistia. (HDC Mobile Store 2013.)

Suurista laitemääristä johtuen Android on saanut kritiikkiä liiallisesta fragmentaatiosta. Tämä haittaa ohjelmoijia, sillä ohjelmoijat joutuvat tukemaan suuria määriä eri resoluutioita, prosessori- ja näytönohjaintehoja ja muita ominaisuuksia.

3.3 Ohjelmointi Androidille

Ohjelmointi Androidille tapahtuu joko Android SDK:lla Java-kielellä tai Android NDK apuohjelmaa käyttäen C tai C++ -kielillä. NDK:n käyttö on rajattu tämän opinnäytteen ulkopuolelle. Googlen SDK:n ohjeissa käytetään Eclipse-ohjelmointiympäristöä, Android SDK tarjoaa pluginin Eclipseä varten. (Android 2013.) Eclipse helpottaa sekä ohjelmointia että ohjelman testausta. Kun Android-emulaattori on määritelty tai oikea laite ajureineen liitetty, ohjelman testaaminen onnistuu nappia painamalla.

Android-ohjelmoinnissa ohjelmat aloitetaan yleensä SDK:n tarjoamalla projektipohjalla, jossa on valmiiksi määriteltynä Android-ohjelman pakollinen `onCreate` −metodi ohjelman pääluokkatiedostoon. Tämä metodi ajetaan kun Android-ohjelma käynnistyy, eikä metodia sen jälkeen enää ajeta. Muita metodeita Android-ohjelmaa ajettaessa ovat `onStart`, joka ajetaan kun ohjelma lähtee käyntiin ja tulee päällimmäiseksi luonnin jälkeen, tässä metodissa on hyvä käynnistää pelin pääsilmukka. `onPause`, ajetaan jos jonkin muun ohjelman osa tulee eteen, esimerkiksi tekstiviesti-ikkuna. Kun `onPause`-metodista palataan, järjestelmä kutsuu `onResume`-metodia, johon voi toteuttaa tarvittavia asioita, kun pause-tilasta palataan. Jos ohjelma joutuu kokonaan taustalle esimerkiksi puhelun tullessa, järjestelmä kutsuu `onStop`-metodia. Tässä metodissa on tarkoitus pysäyttää käynnissä olevat toiminnot, mitkä eivät kuulu taustatoimintoihin. Esimerkiksi pelin pääsilmukka kannattaa pysäyttää, ettei pelaaja häviä peliä kesken puhelun. Päälle menneestä ohjelmasta palauduttua järjestelmä kutsuu ensin `onRestart`-metodia ja tämän jälkeen `onStart`-metodia. (Android 2013.)

Android-projektia luodessa Eclipse tekee myös resurssihakemistot. Näitä hakemistoja ovat: drawable, joihin laitetaan piirrettäviä kuvia, esimerkiksi ohjelman ikoni. Tässä hakemistossa on omat hakemistot eri resoluutioisille näytöille. Layout, joka sisältää eri käyttöliittymätiedostot. Values, johon voi määritellä erilaisia arvoja, kuten ohjelman nimen.

3.4 Peliohjelmointi Androidille ilman frameworkia

Androidin SDK tarjoaa Android-pelikehitykseen Lunar Lander esimerkkipelin. Pelin lähdekoodista voi tarkastella kuinka esimerkiksi törmäyksentunnistus tehdään ilman valmiita metodeja. Esimerkkipelin törmäyksentunnistus on yksinkertainen, katsotaan aluksen mitat ja vertaillaan niitä maalin koordinaatteihin. Yksinkertaisuudesta huolimatta nämä rivit koodissa ovat suhteellisen pitkiä, verrattuna AndEnginen valmiiksi toteutettuun `collidesWith`-metodiin. Lunar Landerissä on käsin ohjelmoidut fysiikka- ja piirtometodit ja se käyttää aluksen liikuttamiseen fyysisiä nuolinäppäimiä, joita ei nykyään puhelimissa juuri ole. Aluksen liikuttamisen toteuttavat metodit vaativat paljolti koodia, jonka lukeminen on hankalaa, sillä muuttujia käytetään useammassa eri metodissa koodia. AndEnginen `AnalogOnScreenControl` luo vastaavanlaisen toiminnallisuuden huomattavasti helpommin luettavalla ja lyhemmällä koodimäärällä. Esimerkkipelissä aluksella ei ole minkäänlaista animointia, alusta pyöritetään, mutta pelin päättyessä aluksen kuva vain vaihtuu. Android SDK tarjoaa kuitenkin `AnimationDrawable`-luokan, jolla tavallisen kehysanimaation voi tehdä. Tämä tapa tehdä animaatio vaatii yhden kuvan per kehys ja koodin lisäksi XML-tiedoston, jossa kuvaresurssit määritellään. AndEngine tarjoaa `AnimatedSprite`-luokan, jolla vastaavanlainen animaatio toteutuu viidellä rivillä koodia, mutta tarvitsee yhden kuvan, jossa kaikki animaation kehykset ovat.

4 ANDENGINE

4.1 Mikä on AndEngine?

AndEngine on avoimen lähdekoodin 2D-peliohjelmointikehys Androidille, se tukee OpenGL ES versioita 1 ja 2. Vaikka AndEngine on pääasiassa tarkoitettu pelikehitykseen, sillä voi myös tehdä ”tavallisia” mobiiliohjelmia, mutta AndEnginen hyödyt jäävät tavallisissa ohjelmissa kovin pieniksi. AndEnginen on kehittänyt alun perin Nicolas Gramlich. Heinäkuussa 2011 Nicolas Gramlich kirjoitti blogissaan liittyvänsä Zyngan mobiilikehitystiimiin ja lupasi, että AndEngine tulee pysymään ilmaisena ja avoimen lähdekoodin projektina. (Gramlich 2011.) AndEngineä voi laajentaa lisäosilla, jotka tarjoavat lisäominaisuuksia AndEnginen perusominaisuuksiin. Näitä lisäosia on mm. fysiikkamoottori ja moninpelijärjestelmä. AndEnginen oma dokumentaatio perustuu lähinnä erilaisiin esimerkkeihin Java-lähdekoodin muodossa mutta AndEnginellä on myös keskustelupalsta, josta löytyy paljon erilaisia hyvin selitettyjä esimerkkejä joilla aloittelevatkin mobiiliohjelmoijat pääsevät helposti alkuun.

4.2 Mitä hyötyä AndEnginen käytöstä on?

AndEnginen tarkoituksena on helpottaa peliohjelmointia Androidille. Pelikehitystä aloittaessa tarvitsee yleensä toteuttaa todella monia yleisiä asioita kuten esimerkiksi skaalautuminen erilaisille resoluutioille, fysiikka, erilaisten asioiden koordinointi ruudulla, taustakuvan suhde muihin esineisiin ja törmäyksentunnistus. Törmäyksentunnistus on yksi tavallisimpia peleissä tapahtuvia asioita ja AndEnginessä on tähän tarkoitukseen muutamia helppokäyttöisiä metodeja, jotta tätäkään perusominaisuutta ei tarvitse itse kirjoittaa. Peleissä, joissa pelihahmoa voi liikuttaa ruudulla, on yleensä näkyvissä tai piilotettuna virtuaaliset ohjaimet, näiden implementointiin AndEngine tarjoaa yksinkertaisen ja helppokäyttöisen ominaisuuden AnalogOnScreenController. Nämä ohjaimet yrittävät emuloida perinteistä peliohjainta.

4.3 AndEnginen ominaisuuksia

AndEngine tarjoaa monia ominaisuuksia valmiina, joita tavallisessa Android SDK:ssa ei ole tai ovat hankalia toteuttaa. Näitä ominaisuuksia ovat mm. AnalogOnScreenControl, Collision Detection, Multiplayer ja ParticleSystem. AnalogOnScreenControl tuo näytölle virtuaalisia ohjaimia, jolla voi esimerkiksi liikuttaa pelaajan hahmoa. Collision Detection tarkoittaa törmäyksen havaitsemista, jota tarvitaan esimerkiksi ammuksen ja vihollisen osumisen toimintoihin. Collision Detection on yksi yleisimpiä peleissä tapahtuvia asioita. Multiplayer tuo mukaan moninpelin, jossa kaksi tai useampi pelaaja voi pelata samaa peliä yhtä aikaa verkkoyhteyden kautta. ParticleSystem helpottaa partikkelien tekoa, kuten esimerkiksi laatikon hajoamisesta tulevia sirpaleita. Animoitujen taustakuvien tekoon on olemassa AutoParallaxBackground, jolla pystyy tekemään liikkuvan loppumattoman taustakuvan pelille.

4.4 AndEnginen käyttöönotto

Ennen AndEnginen asennusta ladataan joko ADT (Android Developer Tools) tai erikseen Eclipse johon asennetaan Android SDK plugin. Kirjoitushetkellä AndEnginen suurin tuettu API Level oli 16, joten Android Build Toolsista piti myös asentaa tämä versio.

AndEngine ei tarjoa valmiiksi käännettyä jar-pakettia, se käännetään käsin. AndEnginen kääntämiseksi tarvitsee ensin kloonata lähdekoodi GitHub-palvelusta käyttäen Git-versionhallintaohjelmistoa. Kloonaus tapahtuu haluamassa hakemistossa komennolla: `git clone https://github.com/nicolasgramlich/AndEngine.git`
Komennon ajamisen jälkeen AndEnginen lähdekoodi löytyy AndEngine-hakemistosta.

Kloonaamisen jälkeen avataan Eclipse, josta valitaan File->Import, tämän jälkeen avautuu ikkuna, josta valitaan General -> Add existing projects into Workspace ja painetaan alhaalta Next-painiketta. Valitse choose root directory kloonattu AndEngine-hakemisto ja paina Finish. Tässä kohtaa Eclipse kääntää AndEnginestä automaattisesti jar-paketin, jonka voi liittää omaan projektiin.

AndEngine otetaan käyttöön Eclipse-projektissa lisäämällä se projektin vaatimiin kirjastoihin samalla tavalla muutkin ulkoiset kirjastot. Projektin ominaisuuksista Java Build Path -kohdassa Libraries-välilehdellä on nappi ”Add external JARs…”, valitse juuri käännetty andengine.jar ja AndEngine on käytössä projektissa.

5 MENETELMÄT

5.1 Ohjelmointiympäristö ja testauslaitteet

Ohjelmointiympäristönä tässä opinnäytetyössä käytetään Eclipseä, jossa on lisäosana Android SDK. Android SDK:n mukana tuleva Android-emulaattori on hyvä apuväline, mikäli oikeaa Android-laitetta ei ole saatavilla. Opinnäytetyön käytännön osuudessa käytetään testilaitteina Samsung Galaxy Spicaa, S:ää, S3:a sekä Nexus 7 -tablettia. Android-kehityksessä on hyvä testata erikokoisilla ja -tehoisilla laitteilla, jotta loppukäyttäjien kokemukset olisivat mahdollisimman samanlaiset.

TAULUKKO 2. Käytettävät testauslaitteet

Laite	Android versio	Resoluutio (vaaka-tasossa)	Prosessori
Samsung Galaxy Spica	2.1 Eclair	480x320	0,8GHz
Samsung Galaxy S	4.1 Jelly Bean	800x480	1GHz ARM Cortex A8
Nexus 7	4.3 Jelly Bean	1280x800	1.3GHz quad-core Cortex-A9
Samsung Galaxy S III	4.1 Jelly Bean	1280x720	1.4GHz quad-core Cortex A9

5.2 Työvälineet

AndEngine -peliohjelmointiprojektin aloittamiseen välttämättömiä työkaluja itse AndEnginen lisäksi ovat Android SDK ja Java-kääntäjä. Pelkästään näiden käyttö ei kuitenkaan ole suositeltavaa, sillä esimerkiksi Eclipsen editoria käyttämällä saa mukaan graafisen käyttöliittymän lisäksi mm. koodin täydennyksen ja dokumentaation. Tämän lisäksi Android SDK tuo Eclipseen Android-ohjelmointia helpottavia aputyökaluja, ku-

ten emulaattorin. Vaikkakin Android SDK:ssa tulee mukana emulaattori, oikean fyysisen puhelimen tai tabletin käyttö erittäin suositeltavaa, sillä emulaattori on huomattavasti hitaampi kuin fyysinen laite. Fyysisen laitteen voi liittää tietokoneeseen laitteen mukana tulleella USB-kaapelilla.

Tämän opinnäytetyön toteutuksessa on käytössä seuraavat työvälineet:
Ohjelmistot: Eclipse, Android SDK, AndEngine
Fyysiset laitteet: tietokone ja taulukossa 2 esitellyt Android-laitteet

5.2.1 Eclipse

Tässä opinnäytetyössä Eclipsellä tarkoitetaan vain Eclipse säätiön kehittämää Java-ohjelmointiympäristöä. Eclipse on Javalla kehitetty Java-ohjelmointiympäristö, joka toimii Windows, Linux ja Mac OS X käyttöjärjestelmillä. Eclipse tukee monia erilaisia lisäosia, tämän opinnäytetyön kannalta merkittävin lisäosa on Android SDK. Eclipse itsessään on erittäin käytännöllinen työkalu kaikenlaiseen Java-ohjelmointiin. Se sisältää hyvät projektityökalut ja sisäänrakennetun Java-konsolin ja debuggerin, jolla pystyy hyvin seuraamaan mitä kehittämäsi sovellus tekee kesken ajon. Debuggeri mahdollistaa ohjelman ajon keskeytyksen ja jatkamisen ennalta valitussa pisteessä. Itse ohjelmointia helpottamassa Eclipsestä löytyy ohjelmakoodin värikoodaus ja koodia kirjoittaessa tapahtuva automaattinen koodin täydennys, josta saa myös JavaDoc-dokumentaation, mikäli sellainen on tarjolla kyseessä olevaan täydennykseen.

5.2.2 Android SDK

Android SDK on Googlen tarjoama ohjelmistokehityspaketti Android-ohjelmointiin. Android SDK:n mukana tulee itse ohjelmakoodien lisäksi kehittämistä helpottavia työkaluja kuten emulaattori jolla voi testata tekemiään sovelluksia, paketointityökalu jolla saa tehtyä ohjelmasta Android-käyttöjärjestelmän ymmärtämän asennuspaketin, dokumentaatio, Android-alustan levykuvat eri versioille ja esimerkkikoodia.

6 ESIMERKKIPELIN TOTEUTUS

6.1 SimpleBaseGameActivity

SimpleBaseGameActivity on AndEnginessä määritelty Activity-luokka, joka on tarkoitettu yksinkertaisen pelin tekemiseen. SimpleBaseGameActivity esittelee kolme pakollista toteutettavaa metodia, jotka ovat: `onCreateEngineOptions`, `onCreateResources` ja `onCreateScene`.

OnCreateEngineOptions-metodi palauttaa `EngineOptions`-objektin, joka määrittelee pelimoottorille tarvittavat muuttujat. Nämä muuttujat asetetaan EngineOptions:n konstruktorissa, ne ovat:

- boolean-arvo onko peli täydessä ruudussa
- ScreenRotation-objekti, joka määrittelee missä suunnassa peli on ja voiko pelin suunta vaihtua kesken pelin
- IResolutionPolicy, joka määrittelee miten ruutu täytetään resoluution mukaan
- Camera-objekti, joka määrittelee ruudulla näkyvää kuvaa. Pelialue voi olla isompi kuin ruudulla näkyvä osuus

OnCreateResouces-metodissa määritellään ennen pelin näyttämistä tarvittavat resurssit kuten äänet, kuvat ja tekstit. Näistä resursseista kerrotaan tarkemmin kappaleessa 6.2 Resurssien lataus.

OnCreateScene-metodissa määritellään itse pelin tai muun aktiviteetin tapahtumia. Aikaisemmat metodit luovat pohjan, tämä metodi määrittelee itse pelin kulkua. OnCreateScene palauttaa Scene-objectin, tähän objektiin liitetään kaikki pelissä olevat asiat, kuten spritet, taustakuvat, painettavat alueet ja muut pelin kulussa tarvittavat asiat.

Yllämainittujen metodien lisäksi voi luoda `onCreateEngine`-metodin, jossa voi halutessaan vaihtaa pelimoottorin tyyppiä. Tässä esimerkkipelissä käytetään LimitedFPSEngineä, joka rajoittaa ruudunpäivitysnopeuden ennalta määritettyyn ylärajaan.

6.2 Resurssien lataus

Resurssien lataus tapahtuu aiemmassa kohdassa mainitussa onCreateResources-metodissa. Tässä esimerkkipelissä käytetään seuraavia resursseja: tekstuuri, ääni ja musiikki.

Tekstuuriresurssin latauksessa tarvitaan kaksi asiaa: bittikarttakuva halutusta grafiikasta sekä alue, johon tekstuuri piirretään. Ensiksi luodaan tekstuuri, tekemällä uusi `BitmapTextureAtlas`-objekti, tämä objekti määrittelee tekstuurin koon ja asetukset. Tämän luokan konstruktori vaatii seuraavat parametrit:

- Engine-objektin tekstuurimanageri
- leveys
- korkeus
- asetukset

Leveys ja korkeus -parametrien pitää olla kahden potenssi, esimerkiksi 64, 128 tai 256. Kun tekstuuri on luotu, voidaan sen avulla luoda alue, johon itse kuva asetetaan. Alueen luonti ja kuvan asetus tekstuuriin tapahtuu luomalla `TextureRegion`-objekti käyttämällä `BitmapTextureAtlasTextureRegionFactory`-luokan staattista metodia `createFromAsset`. Metodille annetaan parametreiksi juuri luotu BitmapTextureAtlas, Context-objekti, polku haluttuun kuvaan, kuvan x- ja y-koordinaatit, joista tekstuurin piirto aloitetaan.

```
bgAtlas = new BitmapTextureAtlas(
this.mEngine.getTextureManager(), 2048, 2048,
TextureOptions.BILINEAR_PREMULTIPLYALPHA);

bgRegion =
BitmapTextureAtlasTextureRegionFactory.createFromAsset(thi
s.bgAtlas, this, "gfx/background.png", 0, 0);
```

Äänien ja musiikin lisääminen on hieman yksinkertaisempaa, ne vaativat vain yhden koodirivin, mutta koodirivi tarvitsee ympäröidä try {} catch {} lohkoilla. Musiikin ja

äänen luomiseksi EngineOptions-objektin pitää sallia äänet ja/tai musiikin. Perusidea ääni- ja musiikkiresurssien lataamisessa on samanlainen kuin tekstuurialueen luonnissa.

```
          try {
          this.bgm =
MusicFactory.createMusicFromAsset(this.mEngine.getMusicManager()
, this, "bgm.mp3");
          this.shootsound =
SoundFactory.createSoundFromAsset(this.mEngine.getSoundManager()
, this, "shot.mp3");
          } catch (IOException e) {
            e.printStackTrace();
          }
```

6.3 Käyttöliittymä

Pelin käyttöliittymä tässä tapauksessa tarkoittaa ensimmäistä interaktiivista ruutua, jonka peli tarjoaa. Tässä ruudussa on yleensä aina Play-nappi ja mahdollisesti napit tuloksille ja asetuksille.

Aloitusruudun pelille voi tehdä joko Androidin tarjoamilla perusominaisuuksilla tai AndEnginellä. AndEngine voi auttaa koristelemaan pelin aloitusruutua, mutta sen suurempia parannuksia AndEngine ei tähän käyttötapaukseen tarjoa.

Päävalikkoon tehdään AndEnginen tarjoamilla `AutoParallaxBackground`- ja `ButtonSprite`-metodeilla hieman graafista ulkonäköä. AutoParallaxBackground:lla luodaan päävalikkoon liikkuva taustakuva ja ButtonSprite:llä tehdään nappi, josta itse peli lähtee käyntiin. Päävalikon taustakuva on koko ruudun kokoinen mustataustainen tähdillä varustettu kuva, se liikkuu kokoajan vasemmalle. Kun kuva päättyy, se alkaa uudelleen oikeasta reunasta lähtien. Play-nappi on staattinen kuva, joka käynnistää pelin.

KUVA 1. Päävalikko.

6.4 Pelin taustagrafiikat

Scene-objektiin on hyvä liittää taustakuva tai ainakin taustaväri. Ilman taustakuvaa tai -väriä ei voida varmistua siitä, mitä taustalla näkyy peliä pelatessa. AndEngine tarjoaa taustakuvan asettamiseen useamman vaihtoehdon:

- Background - tavallinen väritausta
- SpriteBackground - Sprite-tyyppinen staattinen taustakuva
- RepeatingSpriteBackground - staattinen taustakuva, jossa toistuu sama kuva useamman kerran
- AutoParallaxBackground - mahdollisesti moniulotteinen ja animoitu taustakuva

Background-taustavärin asettaminen on erittäin yksinkertaista, parametrit ovat RGB-värit. Esimerkkinä sininen taustaväri:

```
scene.setBackground(new Background(0, 0, 100));
```

SpriteBackground-taustakuvan asettaminen vaatii Sprite-tyyppisen objektin parametriksi.

```
Sprite bg = new Sprite(0, 0, this.bgRegion,
this.getVertexBufferObjectManager());
scene.setBackground(new SpriteBackground(bg));
```

Kappaleessa 6.3 mainittu AutoParallaxBackground mahdollistaa useamman eri taustakuvan samanaikaisen käytön. Nämä taustakuvat voivat olla staattisia tai liikkuvia. Jos käytössä on useampi liikkuva taustakuva, näille voidaan antaa eri liikkumisnopeudet ja saadaan aikaan kolmiulotteinen efekti. AutoParallaxBackground:n konstruktori ottaa parametreiksi RGB-värit sekä päivitysnopeuden. Esimerkissä luodaan mustataustainen AutoParallaxBackground-objekti.

```
AutoParallaxBackground apbg = new
AutoParallaxBackground(0, 0, 0, 5);
```

Kun AutoParallaxBackground-objekti on luotu, siihen tarvitsee lisätä ParallaxEntity-objekteja `attachParallaxEntity`-metodilla. ParallaxEntity:n konstruktori ottaa vastaan parametrit: nopeus ja muoto-objekti. Muoto-objektina käy muun muassa Sprite-luokan objekti, jota tässä esimerkissä käytetään.

```
Sprite bg = new Sprite(0, 0, this.bgRegion,
this.getVertexBufferObjectManager());
apbg.attachParallaxEntity(new ParallaxEntity(-13f,
bg));
```

Tässä vaiheessa taustakuvan luonti on valmis ja se voidaan liittää scene-objektiin.

```
scene.setBackground(apbg);
```

6.5 Pelaajan liikuttaminen

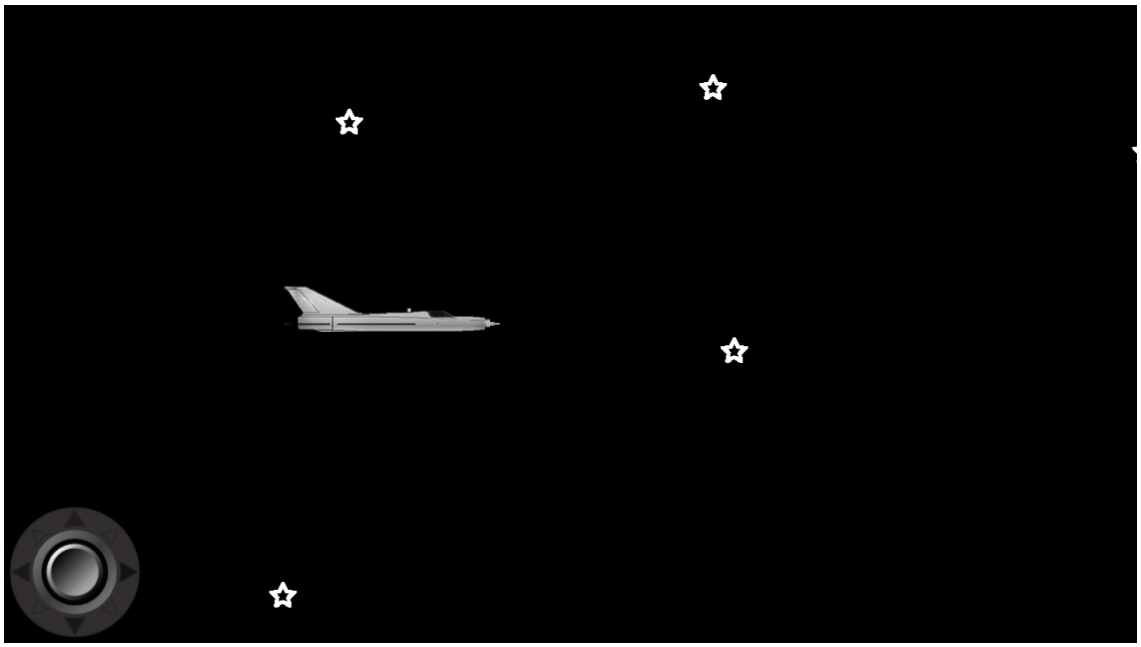
Pelaajan liikuttamisen voi toteuttaa AndEnginen `AnalogOnScreenController`-luokalla. Ensiksi pelaajalle luodaan Sprite-objekti, johon pystyy liittämään erilaisia käsittelijöitä ja liitetään pelaaja sceneen.

```
Sprite player = new Sprite(pTextureRegion.getWidth(),
CAMERA_HEIGHT/2, pTextureRegion,
mEngine.getVertexBufferObjectManager());
scene.attachChild(player);
```

Spriten luomisen jälkeen pelaaja-objektiin voi liittää `PhysicsHandler`:n, joka myöhemmin liitetään AnalogOnScreenControlleriin.

```
PhysicsHandler ph = new PhysicsHandler(player);
Player.registerUpdateHandler(ph);
```

AnalogOnScreenController luo kahdesta tekstuurista koostuvan ohjaintatin ruudulle. Ensimmäinen tekstuuri on ohjaimen pohjakuva, tämä pohjakuva määrittelee millaisella alueella ohjaintattia pystyy liikuttamaan. Kuvassa 2 näkyy pelaajan Sprite (hävittäjä) sekä vasemmassa alanurkassa AnalogOnScreenController. Kun käyttäjä vetää sormellaan ohjaintattia johonkin suuntaan, lähtee hävittäjä liikkumaan kyseiseen suuntaan. Mitä pidemmälle ohjainta siirtää keskipisteestä, sitä nopeammin hävittäjä liikkuu.



KUVA 2. Pelaaja ja AnalogOnScreenController.

AnalogOnScreenController:n konstruktori ottaa parametreina:

- x-koordinaatti, minne ohjain luodaan
- y-koordinaatti, minne ohjain luodaan
- Camera-objekti
- Pohjakuvan textureRegion-objekti
- Tatin textureRegion-objekti
- Päivitysväli
- Aika millisekunteina, milloin painallus lasketaan liikutukseksi, ei painallukseksi
- VertexBufferObjectManager
- IAnalogOnScreenControlListener, jossa määritellään liikutuksen ja painalluksen tapahtumat

`IAnalogOnScreenControlListener` määrittelee mitä tapahtuu, kun ohjainta painetaan tai liikutetaan. Tämän objektin metodeissa toteutetaan pelaajan liikuttaminen

ruudulla. Toteutettavia metodeita on kaksi: onControlChange määrittelee tapahtumat, kun ohjainta vedetään suuntaan tai toiseen, onControlClick määrittelee tapahtumat painettaessa ohjainta. Koodiesimerkki 1:ssä onControlChange-metodissa liikutetaan pelaajaa perusnopeuden ja ohjaintatin asennon perusteella. OnControlClick-metodissa pelaaja ampuu, näin ei tarvitse luoda erikseen nappia ampumista varten. Kun AnalogOnScreenController on täysin luotu, se liitetään scene-objektiin `setChildScene`-metodilla.

```
final AnalogOnScreenControl aosc = new AnalogOnScreenControl(0,
    CAMERA_HEIGHT - this.knobBaseRegion.getHeight(),
    this.camera,
    this.knobBaseRegion,
    this.knobRegion,
    0.1f,
    200,
    this.getVertexBufferObjectManager(),
    new IAnalogOnScreenControlListener() {
        @Override
        public void onControlChange(final BaseOnScreenControl pBaseOnScreenControl, final float pValueX, final float pValueY) {
            ph.setVelocity(pValueX * speed, pValueY * speed);
        }
        @Override
        public void onControlClick(final AnalogOnScreenControl pAnalogOnScreenControl) {
            player.shoot();
        }
    }
);
scene.setChildScene(aosc);
```

KOODIESIMERKKI 1. AnalogOnScreenControllerin luonti.

6.6 Collision detection

Collision detection tarkoittaa kahden asian yhteentörmäyksen tunnistamista. Esimerkkipelissä tämä tarkoittaa pelaajan ampuman ammuksen ja vihollisen törmäystä. Pelaaja, vihollinen ja ammus ovat kaikki Sprite-luokan objekteja. AndEngine:n tarjoamassa Sprite-luokassa on valmiina metodi `collidesWith`. CollidesWith-metodia kutsutaan suoraan Sprite-objektista:

```
sprite.collidesWith(otherSprite);
```

Ennen törmäystarkistuksen toteuttamista pitää toteuttaa Sprite:lle onManagedUpdate-metodi, jossa törmäystarkistuksen voi tehdä. Tätä metodia ajetaan joka kerta, kun ruutu päivittyy. Bullet-luokka on itsekirjoitettu luokka (Koodiesimerkki 2), joka laajentaa Spriteä, näin voi toteuttaa valmiilla ominaisuuksilla olevia Spritejä. Bullet-luokan onManagedUpdate-metodin toteutuksessa näkyy yksinkertainen esimerkki collidesWith-metodin käytöstä.

```
private class Bullet extends Sprite {
    float vx;
    PhysicsHandler ph;
    public Bullet(float sx, float sy, ITextureRegion pTextureRegion,
            VertexBufferObjectManager pVertexBufferObjectManager, float velocity) {
        super(sx, sy, pTextureRegion, pVertexBufferObjectManager);
        // Ammuksen vauhti
        this.vx = velocity;
        ph= new PhysicsHandler(this);
        this.registerUpdateHandler(this.ph);
        // Y-suuntaa ei määritellä, ammukset lentävät suoraan eteenpäin
        this.ph.setVelocity(this.vx, 0);
        this.setTag(BULLET);
    }
    @Override
    protected void onManagedUpdate(final float pSecondsElapsed) {
        if (this.collidesWith(enemy)) {
            this.setVisible(false);
        }
        super.onManagedUpdate(pSecondsElapsed);
    }
}
```

KOODIESIMERKKI 2. Bullet-luokka.

Bullet-luokkaa käytetään, kun ohjainta klikataan vetämisen sijasta. Klikatessa kutsutaan Player-luokan shoot-metodia, joka luo anonyymin instanssin Bullet-luokasta.

```
mEngine.getScene().attachChild(new Bullet(xPos, yCenter,
bulletRegion, this.getVertexBufferObjectManager(), 250));
```

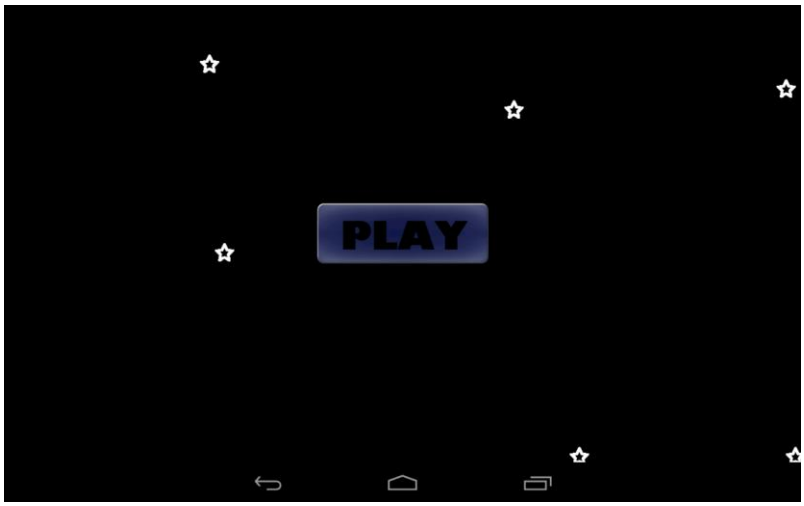
6.7 Testaus

Testaus tapahtui kappaleen 5.1 Taulukko 2 esitetyillä laitteilla. Uusimmat laitteet Samsung Galaxy 3 ja Nexus 7 selviytyivät tehokkuustestauksessa hyvin, FPS pysy tasaisesti 59-60 välillä. Samsung Galaxy S:n FPS heittelehti 44-56 välillä. Testilaitteista heikoin Samsung Galaxy Spica ei pystynyt käynnistämään esimerkkipeliä.

Laite	FPS keskiarvo	FPS pienin	FPS suurin
Samsung Galaxy S 3	59,5	59,1	59,9
Nexus 7	59.85	59,8	59,9
Samsung Galaxy S	49,85	44,0	55,7

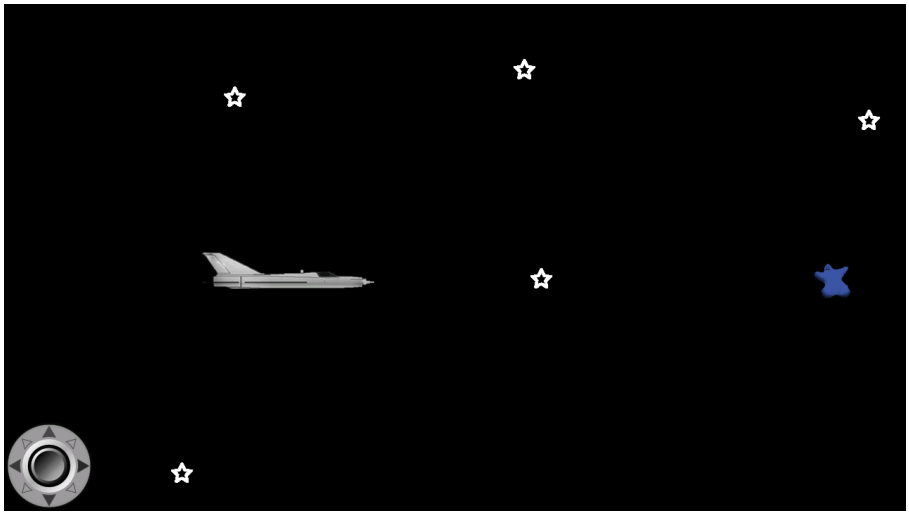
TAULUKKO 3. FPS-tulokset testauksesta.

Ulkonäöllisesti pelin päävalikko näytti laitteissa samalta. Nexus 7 lisäsi omat virtuaalinappinsa pohjaan, mutta muita graafisia eroja ei ollut.

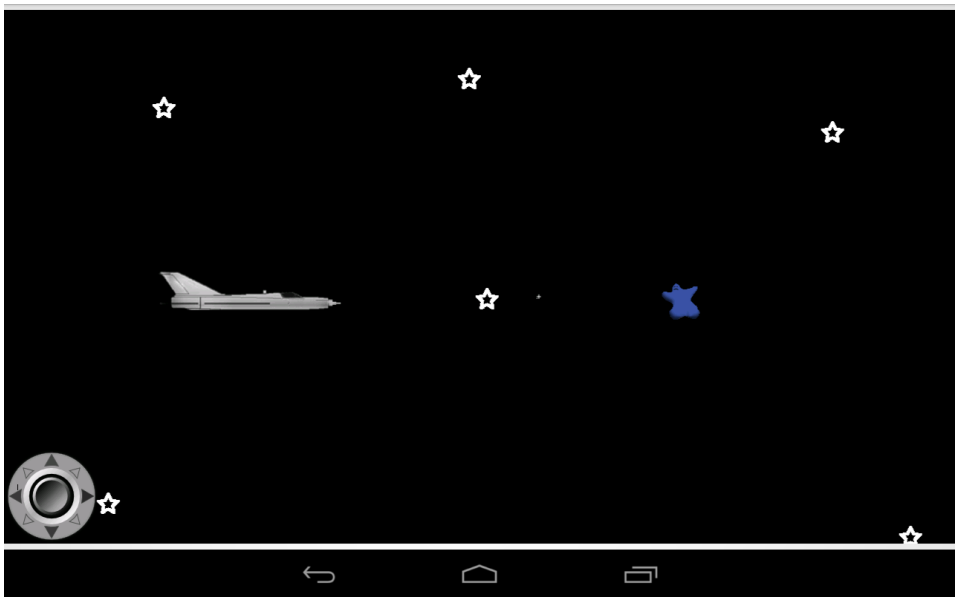


KUVA 3. Päävalikkonäkymä Nexus 7:llä.

Itse pelin aikana laitteiden välillä oli eroja. Samsung Galaxy S 3:lla peli näytti juuri siltä millaiseksi peli oli suunniteltu (kuva 4). Nexus 7 lisäsi alareunaan omat virtuaalinäppäimensä ja niiden lisäksi pelin ylä- ja alareunoihin tuli kaksi ohutta harmaata palkkia (kuva 5). Samsung Galaxy S:llä pelin ylä- ja alareunoihin tuli kaksi valkoista paksumpaa palkkia. Tuntemattomaksi jääneestä syystä Samsung Galaxy S ei myöskään saanut otettua kuvankaappausta pelistä. Pelin IResolutionPolicy oli RatioResolutionPolicy, joka skaalaa pelin kuvasuhteen mukaan. FillResolutionPolicy:llä molemmista laitteista katosi harmaat palkit, mutta tämä aiheutti pientä venymistä grafiikkaan pystysuunnassa. Pienillä kuvasuhde-eroilla tämä ei pelikokemusta haittaa, mutta vielä enemmän neliömäisellä kuvasuhteella grafiikka venyisi jo todella rumaksi.



KUVA 4. Peli Samsung Galaxy S 3:lla.



KUVA 5. Peli Nexus 7:llä.

7 POHDINTA

Nykyaikaisilla Android-laitteilla AndEngine sopii hyvin Android-pelikoodauksen moottoriksi. Uusimmat testilaitteet pyörittivät peliä sulavasti ja nykyaikaisissa laitteissa käytetään pääsääntöisesti 16:19 kuvasuhdetta, joten skaalautuminen ei ole niin suuri ongelma. Vanhemmalla laitteella FPS putosi pelissä hieman, mutta peli oli kuitenkin vielä pelattavissa. Vanhemmille laitteille pelejä suunniteltaessa on hyvä käyttää pienempiä tekstuureja ja miettiä resurssien uudelleenkäyttöä mahdollisimman paljon.

Omiin Android-peliohjelmointikokemuksiini perustuen AndEngine helpotti ohjelmointia todella paljon. Törmäystunnistukset ja säikeet olivat sisäänrakennettuna helposti käytettävissä, omaa säie- tai törmäystunnistuskoodia ei tarvinnut kirjoittaa. Työtä tehdessä löysin Google Play -palvelusta AndEngineExamples -nimisen sovelluksen, joka demonstroi useampia AndEnginen ominaisuuksia, mitä tässä työssä on käytetty. AndEngine:llä voi tehdä todella paljon, tämä työ jäi valitettavasti pääasiassa perusominaisuuksiin.

Aion varmasti tulevaisuudessa tutustua AndEngineen syvemmin ja suosittelen sitä lämpimästi tuleville peliyrittäjille.

LÄHTEET

Nokia. 2013. Publishing your app. Nokia Developer. Luettu 1.4.2013.
http://info.publish.nokia.com/

Google. 2013. Developer Registration. Android Developer. Luettu 1.4.2013.
https://support.google.com/googleplay/android-developer/answer/113468?hl=en

Program Enrollment. Apple Developer. Luettu 1.4.2013.
https://developer.apple.com/support/ios/enrollment.html

Google Play. 2013. Angry birds space. Luettu 15.10.2013.
https://play.google.com/store/apps/details?id=com.rovio.angrybirdsspace.ads

Hildenbrand, Jerry. 2012. Android A to Z: What is Dalvik. Luettu 18.11.2013
http://www.androidcentral.com/android-z-what-dalvik

IDC 2013. Lehdistöjulkaisu. Luettu 19.11.2013.
http://www.idc.com/getdoc.jsp?containerId=prUS24442013

HTC. 2008. Web-arkiston kopio lehdistötiedotteeesta. Luettu 10.10.2013.
http://web.archive.org/web/20110712230204/http://www.htc.com/www/press.aspx?id=66338&lang=1033

Rubio, Justin. 2011. The history of Android. IGN. Luettu 9.4.2012
http://www.ign.com/articles/2011/12/22/the-history-of-android

The Verge Staff. 2013. Android: A Visual History. The Verge. Luettu 19.11.2013
http://www.theverge.com/2011/12/7/2585779/android-history

Google Mobile Blog. 2012. Blogikirjoitus. Luettu 7.8.2012.
http://googlemobile.blogspot.com/2012/02/androidmobile-world-congress-its-all.html

Google. 2013. Android Device Gallery. Luettu 10.10.2013
http://www.android.com/devices/

Samsung. 2012. Samsung Galaxy Note 2. Luettu 1.10.2013.
http://www.samsung.com/global/microsite/galaxynote/note2/spec.html?type=find

HDC Mobile Store. 2012. Luettu 1.10.2013.
http://hdcmobile.com/hdc-galaxy-note-2.html

Android. 2013. Developer Tools. Luettu 2.10.2013.
http://developer.android.com/tools/index.html

Android. 2013. Activities. Luettu 19.9.2013.
http://developer.android.com/guide/components/activities.htm

Android. 2007. LunarView.java. Luettu 20.10.2013

Gramlich, Nicholas. 2011. I’m joining Zynga Mobile! Luettu 1.4.2012. http://www.andengine.org/blog/2011/07/i-am-joining-zynga-mobile/