VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Janne Rönnqvist

Kauppapaikkojen integrointi usean alustan peliin

Tietotekniikka
2014

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Janne Rönnqvist
Opinnäytetyön nimi	Kauppapaikkojen integrointi usean alustan peliin
Vuosi	2014
Kieli	suomi
Sivumäärä	**33+17 liitettä**
Ohjaaja	Timo Kankaanpää

Työn lähtökohtana on tehdä usealle eri alustalle peli, joka kaupallistetaan mainonnan ja pelin sisäisen myynnin avulla. Pelin luomiseen on käytetty Libgdx pelimoottoria. Työ on tehty pääosin Java-kielellä, mutta eri alustojen kanssa on käytetty myös PHP, HTML, CSS, XML ja Javascript -tekniikoita. Peli ohjelmoitiin Eclipse ohjelmointiympäristöä käyttäen. Opinnäytetyön pääpaino on kauppapaikkojen integroinnissa peliin.

Pelinteko aloitettiin suunnittelusta. Varsinainen peli-idea oli keksitty jo vuosia ennen projektin aloittamista, joten suunnittelu voitiin aloittaa pienemmistä yksityiskohdista. Peliruudun suunnittelun jälkeen aloitettiin ohjelmointi ja muut valikot suunniteltiin varsinaisen pelin ollessa valmis.

Ohjelmointi aloitettiin peliruudun tekemisestä. Ensiksi piirrettiin komponentit ruudulle, jonka jälkeen niille lisättiin erilaisia ominaisuuksia. Peliä testattiin samalla, kun siihen lisättiin uusia ominaisuuksia. Toimivan peliruudun luomisen jälkeen, peliin luotiin valikot ja muut pienemmät yksityiskohdat.

Pelin julkaiseminen aloitettiin verkkoselaimessa pyörivästä versiosta, koska sen julkaiseminen vaati vähiten työtä. Julkaisun jälkeen peliin lisättiin Amazonin maksujärjestelmä ja se julkaistiin Amazonin kauppapaikalle. Googlen maksujärjestelmän integroinnin jälkeen, peli julkaistiin vielä Google Play versio.

Projekti kokonaisuudessaan oli onnistunut. Kaikki tavoitteet saavutettiin ja peliä tehtäessä kohdatut ongelmat saatiin ratkaistua. Pelistä saatu palaute on ollut pääosin positiivista.

Avainsanat Java, Kauppapaikka, Android, peli

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Janne Rönnqvist
Title	Marketplace Integration for a Multiplatform Game
Year	2014
Language	Finnish
Pages	**33+17 Appendices**
Name of Supervisor	Timo Kankaanpää

The purpose of this thesis was to create a cross-platform game. The game was commercialized through advertising and virtual item sales. The game was created with the Libgdx game engine and programmed with Java, but PHP, HTML, CSS, XML and Javascript languages were also used. Eclipse was used as the main development platform. The main focus of this thesis is on the marketplace integration.

The game creation began with designing. The main game idea was invented years before the actual project began, so the designing process was focused on the smaller details. After designing the game screen the programming phase started. Other screens, such as menus were designed afterwards.

The programming phase started by creation of the game screen. First the game components were drawn to the screen and then abilities for every game component were added. Menus and other smaller details were added after functioning game screen.

Web-browser version of the game was published first, because it required less work than the other versions. After the first release the Amazon payment system was added and then the game was released to the Amazon marketplace. The Google wallet payment system was integrated after the second release and then the game was released to the Google Play marketplace.

The project was a success. Every goal was reached and every problem on the game creation was solved. The feedback from the game has been positive.

Keywords Java, Store, Android, game

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

LIITELUETTELO

1 JOHDANTO

Opinnäytetyön lähtökohtana on tehdä peli, joka tullaan julkaisemaan usealle eri alustalle. Peli tulee olemaan tyypillinen ”Match-3”-genren pulmapeli, joka sisältää useita erilaisia ratoja. Pulmapeli tullaan ohjelmoimaan Java –ohjelmointikielellä, käyttäen apuna Libgdx-pelimoottoria. Ratojen tekemisen helpottamiseksi tullaan tekemään myös ns. rataeditori, Javan Swing-komponentteja hyväksikäyttäen.

Peli tullaan julkaisemaan Chrome-verkkoselaimelle, Google Play:tä tukeville laitteille, sekä Amazonin kauppapaikalle. Tämän jälkeen peli kaupallistetaan mainosten ja pelin sisäisten kauppapaikkojen avulla. Työssä keskitytään maksujärjestelmiin, niiden integrointiin, sekä eri kauppapaikkojen testaamiseen.

Pelin sisäisen kauppapaikan integrointi on jokaiselle alustalle erilainen. Amazon tarjoaa omalle kauppapaikalleen oman maksujärjestelmänsä, kun taas Googlessa käytetään Google Play -kauppapaikkana. Sen sijaan verkkoselaimessa käytettävään Chrome Web Store julkaisuun tarvitaan PayPal-maksujärjestelmää.

Yksi päätavoitteista on saada pelin sisälle toimiva ja helppokäyttöinen maksujärjestelmä. Ideaali järjestelmä tuottaa mahdollisimman vähän vaivaa kehittäjälle integroinnin jälkeen. Tavoitteena on täten saada täysin automatisoitu järjestelmä, jossa vain kolmannet osapuolet huolehtivat luottokortti- ja laskutusasioista.

2 TYÖN TAUSTA

Pelin idea on kehitetty vaihto-opiskelukaudella Thaimaassa, keväällä 2012. Maan matkustuskulttuuriin kuului pitkät bussimatkat ja mobiilipelaaminen oli yksi ajanviettotavoista. Lukuisten bussissa vietettyjen tuntien aikana vaihto-oppilaiden suosikkipeliksi muodostui eräs ratapohjainen ”Match-3” -genren timanttipeli, jota pelattiin vuorotellen ja matkat sujuivat huomattavasti joutuisammin.

Idea pelin tekemisestä syntyi ajatuksesta, jos taidot riittäisivät samankaltaisen pelin kehittämiseen. Vaihto-opiskelukauden aikana osaaminen pelin ohjelmoimisesta oli todella vähäistä, joten projekti siirrettiin taka-alalle odottamaan, lähinnä sen haastavuuden takia. Eri projektien ohella saatujen taitojen kehittymisen myötä pelin idea kehittyi kuin itsestään ja projekti oli mahdollista saada käyntiin keväällä 2013.

Pelin tekemistä varten täytyi opiskella Java ja PHP –ohjelmointikieltä, sekä Google Analytics -analytiikkatyökalun käyttöä. Tämän lisäksi myös Libgdx-pelimoottorin hyödyntäminen tuli täysin uutena asiana, jonka opiskelemiseen kului paljon aikaa.

3 KÄYTETYT TEKNIIKAT

3.1 Android

Android julkistettiin vuonna 2005, kun Google osti Android Inc. -nimisen pienen startup-yrityksen. Tästä seurasi suuri määrä huhuja, joiden mukaan Google olisi siirtymässä jo käynnissä olevaan mobiilikilpailuun. Huhut loppuivat, kun Google julkaisi Android 1.0 version, joka haastoi Applen iOS, Windowsin Windows Phone 7 ja Blackberryn Blackberry OS mobiilikäyttöjärjestelmät. Android käyttöjärjestelmä on kasvanut räjähtävällä vauhdilla ja ajan myötä siitä on kasvanut markkinoiden johtava mobiilikäyttöjärjestelmä.

Kuvio 1. Android logo

Android -käyttöjärjestelmä käyttää Linux kernel versioita 2.6 ja 3.x. Käyttöjärjestelmä perustuu avoimeen lähdekoodiin, joka tekee siitä helposti lähestyttävän eri mobiilaitevalmistajien näkökulmasta. Laitevalmistajat pystyvät käyttämään sitä eri hintaluokan laitteisiin ja muokkaamaan sitä haluamansa tyyliseksi. Android ei

ole pelkästään korkeatehoisten mobiililaitteiden käyttöjärjestelmä. Se on ilmainen niin kaupalliseen, kuin myös ei-kaupalliseen tarkoitukseen.

Android ei ole pelkästään Linux käyttöjärjestelmän versio mobiililaitteille. Sovelluksia kehittäessä, on todennäköisempää että Linux kernel ei tule vastaan. Kehittäjän näkökulmasta ohjelmointi tapahtuu Java-ohjelmointikielellä, jonka alla pyörii Linux kernel. /1/

3.2 Java

Java on Sun Microsystemsin kehittämä alustariippumaton olio-ohjelmointikieli. James Gosling, Mike Sheridan ja Patrick Naughton aloittivat Java projektin kesäkuussa 1991. Java oli alunperin tarkoitettu käytettäväksi interaktiivisten televisioiden kanssa, mutta sen todettiin olevan liian kehittynyt ohjelma kaapelitelevisioihin. Kieltä kutsuttiin aluksi nimellä Oak, erityisesti puulajikkeen innoittamana. Myöhemmin se uudelleen nimettiin Javaksi, Java kahvipavun mukaan. Gosling suunnitteli sisällyttävänsä kieleen virtuaalikoneen ja kielen täytyi olla C++ kielen kaltainen.

Ensimmäinen julkinen versio Javasta tuli markkinoille vuonna 1995. Sitä mainostettiin myyntilauseella ”Write Once, Run Anywhere”, joka kuvasi Javan monipuolisuutta toimia erilaisilla alustoilla. Hieman tämän jälkeen suurimmat Webselaimet alkoivat tukea Java-appletteja. Tästä johtuen Java yleistyi nopeasti ja siitä julkaistiin Java 2 versio joulukuussa 1998.

Java -koodi on mahdollista kääntää virtuaalikoneen avulla. Ohjelmointikoodi kirjoitetaan samalla tavalla joka alustalle. Virtuaalikone kääntää kirjoitetun koodin halutulle alustalle sopivaksi. /2/

3.3 PHP

PHP -lyhenne tulee sanoista PHP hypertext preprocessor ja se on erityisesti webpalvelimen ohjelmointikieli. PHP kieltä käyttävät websivustot tunnistaa ”.php”-

päätteestä. Päätettä lukuun ottamatta, PHP-kieltä hyödyntävät web-sovellukset näyttävät samalta kuin normaalit HTML-sivustot. Pelkän kuvauksen lisäksi, PHP:n avulla on mahdollista lisätä sivustolle tietokantatuen tai muita loogisia toimintoja.

Vuonna 1994 Rasmus Lerdorf kehitti sarjan pieniä työkaluja web-kehitykseen. Niihin kuuluivat vieraskirja, vierailijalaskuri ja muita kotisivu-elementtejä. Lopulta hän yhdisti nämä työkalut ja lisäsi tietokantatuen. Tämä paketti julkaistiin nimellä PHP/FI.

Paketin julkaisun jälkeen, avoimen lähdekoodin hengessä kehittäjät ympäri maailmaa alkoivat kehittää PHP/FI pakettia eteenpäin. Vuoteen 1997 mennessä, yli 50 000 Internet-sivustoa käytti PHP/FI- pakettia erilaisten ongelmien ratkaisemiseksi.

Ajan myötä PHP/FI paketti oli kehittynyt siihen pisteeseen, että kehitystyö oli siirtynyt yhden henkilön projektista kokonaisen tiimin työksi. Tähän aikaan Lerdorfin kanssa pääkehitystyössä olivat mukana Zeev Suraski ja Andi Gutmans. Heidän avullaan luotiin versio 3.0 parserista. Lopullinen versio PHP 3.0 julkaistiin kesäkuussa vuonna 1998. Se sisälsi tuen usealle eri alustalle, alkuperäisen Linux-tuen lisäksi. Paketti sisälsi myös tuen usealle eri tietokannalle ja samalla tuen SNMP ja IMAP protokollille. Opinnäytetyötä varten käytössä on ollut versio PHP 6.0. /3/

3.4 HTML

HTML tulee sanoista HyperText Markup Language. HMTL on kuvauskieli, jota käytetään web-sivustojen luomiseen. HTML -kielessä käytetään erilaisia elementtejä, joiden avulla kuvataan sivuston rakennetta. Esimerkiksi otsikkoa kuvataan <h1> - elementillä ja kuvaa <img> - elementillä. Web-selain kääntää kuvauskielen ihmissilmälle tutuksi web-sivustoksi.

Vuonna 1980 CERNin fyysikko Tim Berners-Lee ehdotti ja kehitti prototyypin järjestelmästä CERNin kehittäjien dokumenttien jakamiseen. Berners-Lee määritteli HTML kielen ja teki selain-serverijärjestelmän vuoden 1990 lopussa.

Ensimmäinen virallisesti julkaistu kuvaus HTML kielestä oli nimeltään ”HTML Tags”, ja se julkaistiin vuonna 1991 Berners-Leen toimesta. Se kuvasi 18 erilaista elementtiä ja oli erittäin yksinkertainen versio HTML kielestä.

Nykyään HTML on perusta kaikille web-sivustoille ja se toimii standardina web-sivuja tehtäessä. Tätä työtä tehtäessä uusin versio HTML kielestä oli versionumero 5. HTML5 toi mukanaan paremman tuen dynaamisille websivustoille ja graafisen rajapinnan pelien ja muun grafiikan esittämiseen. /4/

3.5 Sovellukset ja työkalut

3.5.1 Eclipse

Eclipse on avoimeen lähdekoodiin perustuva ohjelmointityökalu. Eclipse on yleisesti käytetty Java-ohjelmoinnin parissa, mutta Eclipsestä löytyy myös C++-ohjelmointiin suunnattu versio.

Tässä työssä Eclipseä käytettiin itse pelin tekemiseen. Se valittiin, koska Eclipse on yleisesti käytetty Android ohjelmoinnin parissa. Pelin valmistumista nopeutti hieman Eclipsen entuudestaan tunteminen, sillä se on ollut käytössä myös aikaisemmissa projekteissa. /5/

3.5.2 Libgdx

Libgdx on Java-pohjainen pelienkehitystyökalu. Se perustuu avoimeen lähdekoodiin ja on ilmainen niin kaupalliseen kuin ei-kaupalliseen käyttöön. Se helpottaa kehitystyötä, kun tavoitteena on luoda tuote monelle eri alustalle.

Libgdx:n avulla on mahdollista saada käännettyä ohjelmointikoodi Android, Windows, Linux, Mac OS X, Blackberry ja IOS – alustoille. Libgdx tukee myös selaimessa pyöriviä WEBGL ja Java Applet tekniikoita. Libgdx:n ympärillä pyörii aktiivinen yhteisö, jonka avulla erilaisten ongelmien ratkaisu helpottuu. Libgdx:n käytön opiskelun jälkeen, pelin kehitystyö usealle alustalle on onnistunut helpommin. /6/

3.5.3 Soomla

Soomla on kirjasto, jonka tarkoitus on helpottaa Android-pelin tai sovelluksen sisäisten myyntien integrointia. Soomlasta löytyy tuki seuraaville alustoille: Android, IOS, Unity 3D ja Cocos 2d-X. Soomla tarjoaa myös valmiit teemat ja grafiikat kauppapaikoille. Tässä työssä käytettiin ainoastaan Soomlan tarjoamaa kirjastoa Androidille.

Työtä tehdessä Soomla kirjastosta ei löytynyt tukea Amazonin kauppapaikalle. Tuki kyseiselle kauppapaikalle tuli pelin julkaisun jälkeen. Soomlan peruskirjasto on ilmainen, mutta sisältää myös maksullisia ominaisuuksia /7/

4 PELIN MÄÄRITTELY

Pelin ideana on yhdistää kolme tai useampi samanlaista timanttia vaaka- tai pystytasoon. Kombinaation syntyessä timantit tuhoutuvat ja pelaaja saa pisteitä. Peli tulee sisältämään ratoja, jotka sisältävät erilaisia haasteita. Pelin sisälle luodaan valuutta, jolla pelaaja pystyy helpottamaan etenemistään pelissä ja mahdollistaa pelaajan poistamaan pelissä olevat mainokset. Erilaisten haasteiden suorittaminen edellyttää valuutan jakamista pelaajille. Tämän lisäksi pelaaja voi myös halutessaan ostaa lisää valuuttaa peliin integroidun kauppapaikan kautta oikealla rahalla.

Peli ei saa sisältää yhtään kriittistä ohjelmointivirheittä, jotka voisivat pilata pelaajan kokemuksia pelistä. Sen tulee toimia vaatimusmäärittelyiden mukaisesti (taulukko 1). Valikoiden on oltava yksiselitteiset ja helppokäyttöiset. Pelin kulku selviää käyttötapakaaviosta. (kuva 3.)

Graafinen puoli hankitaan ulkoiselta freelancerilta. Hankittaviin grafiikkoihin kuuluvat erilaiset efektit, timantit ja valikoissa olevat painikkeet. Yksinkertaisimmat grafiikat on piirretty itse. Pelin graafinen puoli selviää alla olevasta kuvasta. (kuva 2.)

Kuvio 2. Peliruutu

4.1 Vaatimusmäärittely

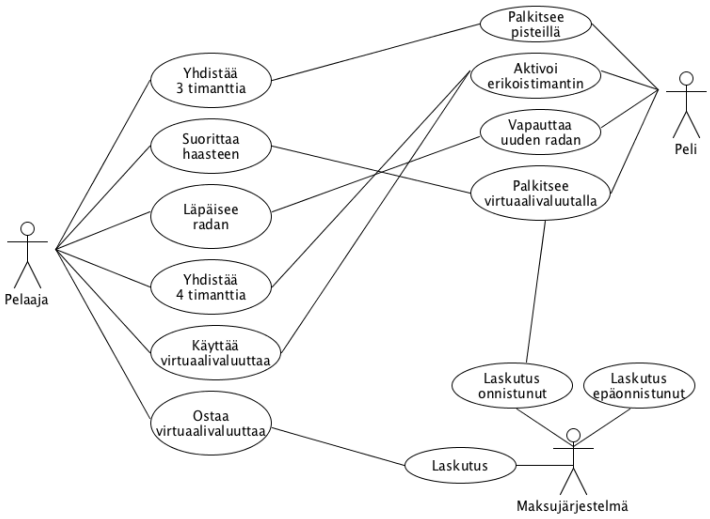
ID	Selite	Prioriteetti
V1	Siirrettävät jalokivet	1
V2	Kolmen jalokiven vaaka tai pystykombinaatio jalokivet tuhoutumaan, jolloin peli lisää automaattisesti uudet	1
V3	Erilaiset rataelementit ja niiden toiminta	1
V4	Radan lukeminen tekstitiedostosta	1
V5	Erilaiset valikot ja valintaikkunat	1
V6	Tehosteet ja äänet	2
V7	Mainokset	2
V8	Kauppapaikkojen integrointi	2
V9	Pelin sisäinen valuutta ja palkinnot	2

Taulukko 1. Vaatimusmäärittely

4.2 Pelin rakenne

4.2.1 Käyttötapakaavio

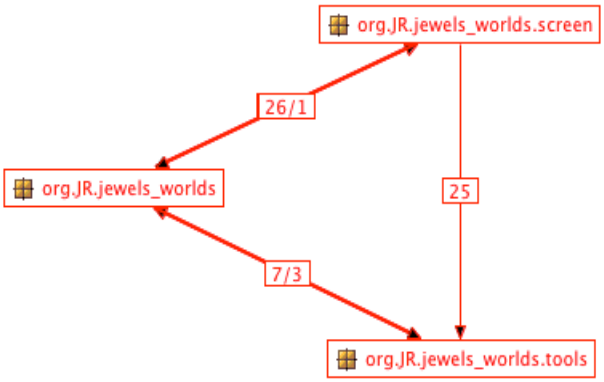
Käyttötapakaaviosta näkee pelin sisältämiä tyypillisiä käyttötilanteita. Kaavio on jaettu kolmeen eri osapuoleen: pelaajaan, peliin ja ulkoiseen maksujärjstelmään.



Kuvio 3. Käyttötapakaavio

4.2.2 Pakettikaavio

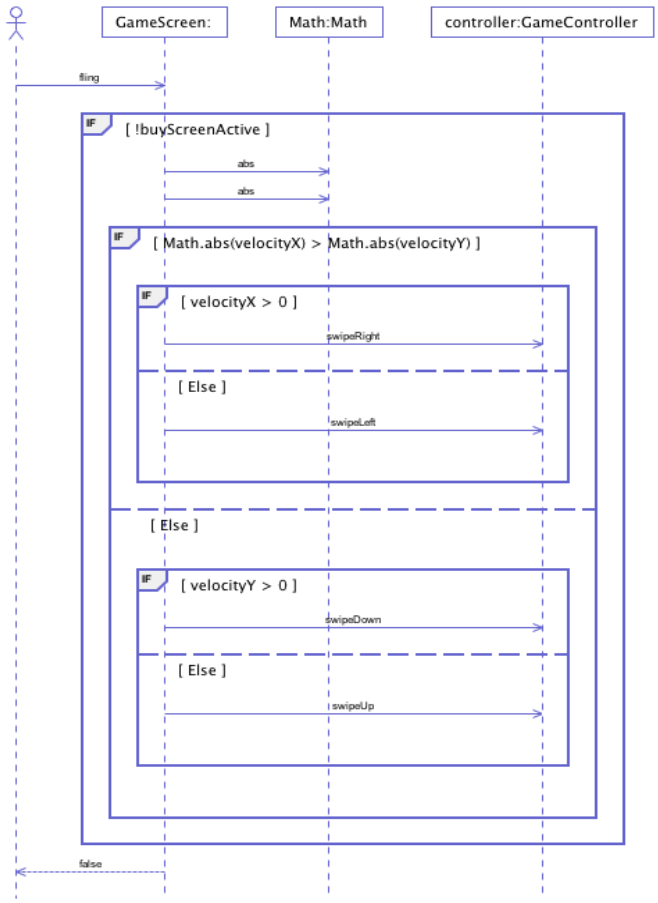
Pakettikaaviosta näkee pelin sisältämät paketit. Screen-paketti sisältää erilaiset peliruudut. Tools-paketti sisältää pelin käyttämät pienet työkalut, kuten tallennus-luokan. Jewels_worlds-paketti sisältää erilaisia pieniä pelikomponentteja.



Kuvio 4. Pakettikaavio

4.2.3 Sekvenssikaavio ruudun koskettaminen

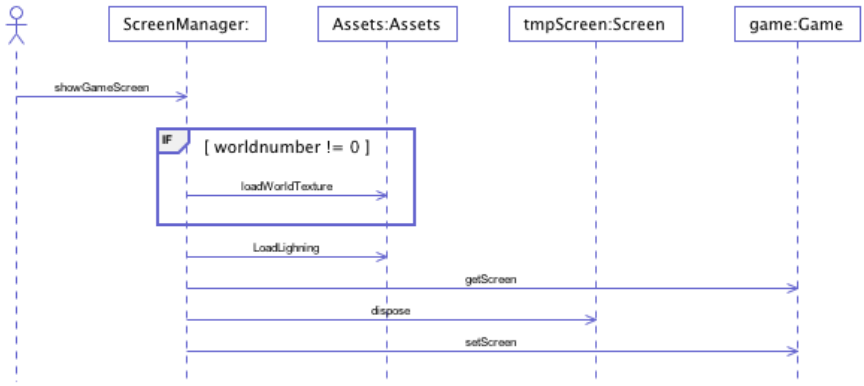
Kaaviosta ilmenee ruudun koskettamisen aikana tapahtuvat toiminnot. Peliruutu tarkistaa onko kosketusliike vaaka vai pystysuuntaista ja sen suunnan. Sen jälkeen liikkeen parametrit ilmoitetaan pelikontrollerille, joka siirtää timantteja tarvittaessa.



Kuvio 5. Sekvenssikaavio ruudun koskettaminen

4.2.4 Sekvenssikaavio peliruudun näyttäminen

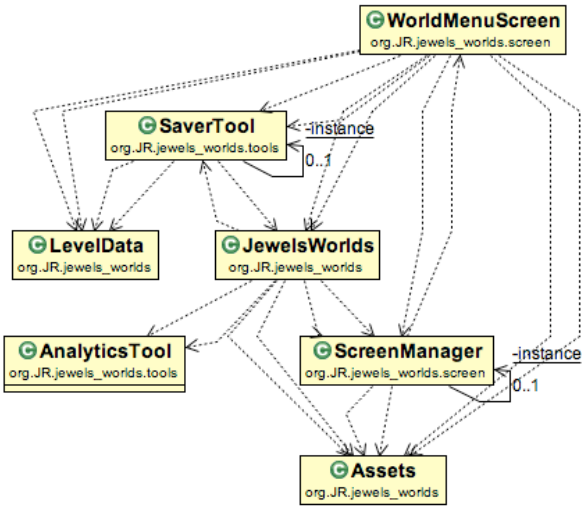
Kaaviosta ilmenee peliruudun näyttämiseen käytettävät luokat. ScreenManager-luokka lataa tarvittavat komponentit ja luo uuden ruudun, jonka jälkeen se poistaa edellisen ruudun muistista.



Kuvio 6. Sekvenssikaavio peliruudun näyttäminen

4.2.5 Maailma-valikon käyttämät luokat

Kuvasta ilmenee maailma-valikon näkyessä käytössä olevat luokat. Selitykset eri luokille löytyvät alla olevasta taulukosta. (Taulukko 2.)



Kuvio 7. Maailma-valikon käyttämät luokat

4.2.6 Luokkien selitykset

Paketti	**Luokka**	**Selitys**
org.JR.jewels_worlds	Assets	Luokka eri grafiikoille
org.JR.jewels_worlds	Block	Peliobjekti
org.JR.jewels_worlds	DestroyedBars	Peliobjekti
org.JR.jewels_worlds	Effect	Peliefekti
org.JR.jewels_worlds	GameController	Pelinydin
org.JR.jewels_worlds	InterfaceBlock	Käyttöliittymäolio
org.JR.jewels_worlds	Jewel	Peliobjekti
org.JR.jewels_worlds	JewelsWorlds	Pelinaloitusluokka
org.JR.jewels_worlds	Level	Rataobjekti
org.JR.jewels_worlds	LevelData	Ratatiedot
org.JR.jewels_worlds	Text	Tekstiobjekti
org.JR.jewels_worlds	Timer	Ajastin
org.JR.jewels_worlds	WorldController	Radan lataus
org.JR.jewels_worlds.screen	AchievementScreen	Haasteruutu
org.JR.jewels_worlds.screen	GameScreen	Pääpeliruutu
org.JR.jewels_worlds.screen	InstructionsScreen	Ohjeruutu
org.JR.jewels_worlds.screen	LevelSelectScreen	Ratavalikko
org.JR.jewels_worlds.screen	MainMenuScreen	Aloitusvalikko

org.JR.jewels_worlds.screen	WorldMenuScreen	Maavalikko
org.JR.jewels_worlds.screen	StoreStage	Kauppapaikkaruutu
org.JR.jewels_worlds.screen	ScreenManager	Valikkojen hallinta
org.JR.jewels_worlds.tools	AchievementUnlocker	Haastepalkitsija
org.JR.jewels_worlds.tools	AdvertisingPlatform -Resolver	Mainonta
org.JR.jewels_worlds.tools	AnalyticsTool	Analytiikkatyökalu
org.JR.jewels_worlds.tools	GoogleInterface	Verkkotilastot ja haasteet
org.JR.jewels_worlds.tools	HTML5KeyValidator	Selain kauppapaikka
org.JR.jewels_worlds.tools	InAppPurchasePlatfo -rmResolver	Kauppapaikkatyökalu
org.JR.jewels_worlds.tools	InAppPurchaseTool	Kauppapaikkatyökalu
org.JR.jewels_worlds.tools	SaverTool	Tiedon tallentaminen
org.JR.jewels_worlds.tools	SoundPlayer	Äänityökalu

Taulukko 2. Luokkien selitykset

5 KAUPPAPAIKKOJEN SUUNNITTELU

5.1 Google Play

Google Playn mobiiliversiohin kauppapaikka integroidaan suoraan pelin sisälle. Valinnanvaraa maksujärjestelmän valitsemiseen ei jää, sillä Googlen kauppapaikkasäännöissä lukee, että maksujärjestelmänä tulee käyttää Googlen omaa Google Wallet – järjestelmää. Työssä käytetään Soomla kirjastoa, Google Walletin integ-

roimisen helpottamiseksi. Julkaisua ja maksujärjestelmän integrointia varten tarvitaan oikeudet Google Play julkaisuun. Kehittäjän tili maksaa Google Playhin 25 dollaria, joka on kertaluontoinen.

Peliin tullaan tekemään neljä eri hintaista pakettia, joista pelaaja saa henkilökohtaisesti valita itselleen parhaimman sopivimman vaihtoehdon. Kalleimmalla paketilla voi saada yhdellä dollarilla eniten pelin sisäistä valuuttaa, kun taas halvimmalla vähiten.

5.2 Amazon Appstore

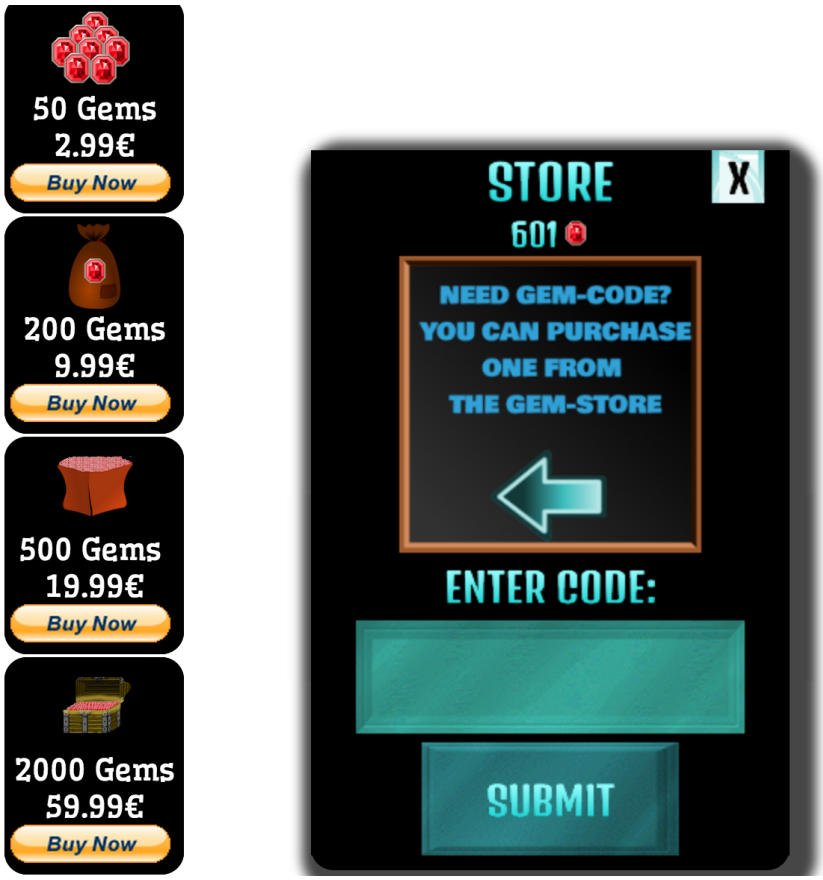
Amazon Appstoren julkaisu on graafisesti identtinen Google Play versioon. Valuuttapaketit ovat samanhintaisia ja kauppapaikka näyttää samalta. Ainoana erona voidaan pitää taustalla pyörivää maksujärjestelmää. Amazon vaatii, että kauppapaikalla käytetään myös Amazonin omaa maksujärjestelmää. Tästä maksujärjestelmästä on hyvä dokumentaatio ja se on kehitetty myös helposti integroitavaksi. Amazonin Appstoren kehittäjätili on ilmainen ja sen rekisteröiminen on yksinkertaista

Kuvio 8. Mobiilipelin sisäinen kauppa

5.3 Chrome Web Store

Kauppapaikan integrointi oli hieman monimutkaisempaa verkkoselain versioon. Suoraan peliin tehtävän maksujärjestelmän integrointi olisi tullut liian vaikeaksi, sillä käytössä ei ollut natiivi web-sovellus. Kauppapaikka tehtiin pelin ulkopuolelle, käyttäen PayPal-maksujärjestelmää. Peliä käytettäessä kauppapaikka on koko-ajan pelaajan nähtävissä. (kuvio 9.)

Valuuttapaketit ovat samanhintaisia, kuin mobiilipelissäkin. Käyttäjän ostaessa valuuttapaketin, sähköpostiin ilmestyy avainkoodi, jonka avulla on mahdollista lunastaa valuuttapaketti pelin sisältä. Oston tapahtuessa PayPal-järjestelmä lähettää ulkoiselle serverille ilmoituksen, jonka ilmoituksesta PHP-ohjelmointikoodi poimii käyttäjän sähköpostin ja ostetun paketin tiedot. Ohjelmointikoodi luo tietojen avulla avainkoodin ja lisää sen sähköpostiviestiin, jonka se lähettää käyttäjän sähköpostiosoitteeseen.



Kuvio 9. WEBGL- version kauppapaikka

6 KAUPPAPAIKKOJEN TOTEUTUS

6.1 Google Play

Kauppapaikkaa varten luodaan valikko, jossa on kolme tekstikenttää otsikkoa ja ohjetekstejä varten. Tekstikenttien lisäksi valikossa on viisi painiketta, yksi kauppapaikan sulkemista varten ja neljä eri pakettien ostoa varten. (kuvio 8.)

Hankitulle Google Play kehittäjätilille luodaan uusi peli, jotta on mahdollista luoda halutut paketit ja saada mahdollisuus vielä testata maksutapahtumia. Pelin ei tarvitse olla valmis tai julkaistu vielä tässä vaiheessa.

Tämän jälkeen ladataan Soomla-kirjasto ja lisätään se osaksi projektin kirjastoja. Soomlan integrointia varten täytyy tehdä Assets-luokka (liite 10.1), joka sisältää tiedot myytävistä paketeista. Assets-luokan lisäksi on luotava tapahtumakuuntelija (liite 10.2), jonka avulla on mahdollista tarkkailla maksutapahtumia ja palkita käyttäjä maksutapahtuman sattuessa. Näiden kahden luokan luomisen jälkeen Soomlan integrointi on helppoa.

Kyseisten luokkien lisäksi pitää pelin alussa ottaa Soomla käyttöön komennolla:

```
StoreController.getInstance().initialize(new JewelsStoreAs-
set(),"GooglePlay Avain","Soomla_avain");
```

Google Play -avaimen voi saada kyseisen kehittäjätilin kautta, kun taas Soomla avaimen saa kehittäjä itse määritellä. Soomla-kirjastolle on ilmoitettava Soomlan käyttöönoton jälkeen, koska käyttäjä avaa ja sulkee kauppapaikan.

Kauppapaikan avaamiseen käytetään komentoa:

```
StoreController.getInstance().storeOpening();
```

Kauppapaikan sulkeminen ilmoitetaan komennolla:

```
StoreController.getInstance().storeClosing();
```

Jokaisen paketin ostopainikkeen kuuntelijametodiin lisätään komento paketin ostoa varten.

```
StoreInventory.buy("Paketin_avain");
```

Tämän jälkeen lisätään vielä laskutusoikeus ja laskutuksen luokkamäärittelyt AndroidManifest.xml tiedostoon seuraavasti:

```
<uses-permission android:name="com.android.vending.BILLING" />

<activity
            an-
droid:name="com.soomla.store.StoreController$IabActivity"
            an-
droid:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen" />
```

Lisäysten jälkeen on mahdollista aloittaa Google Play-version testaaminen. /8/

6.2 Amazon Appstore

Kauppapaikkaa varten käytetään Google Play-versiota varten luotua valikkoa. Amazon Appstoreen luodaan kehittäjätili, jonne lisätään uusi peli ja pelin sisälle myytävät paketit Google Play -version tavoin.

Amazon Appstore -versiota varten tarvitaan observer-luokka (liite 10.3). Siihen saa loistavan pohjan Amazonin dokumentaation kautta. Pelaajan palkitseminen suoritetaan luokan onPurchaseResponse-metodissa tarkistamalla paketin avain.

Observer-luokan lisäksi pelin alussa tulee ottaa Amazon-maksujärjestelmä käyttöön komennolla:

```
PurchasingManager.registerObserver(new AmazonIAPobserver(mActivity));
```

Komennolla rekisteröidään edellä mainittu observer-luokka kirjaston käyttöön. Sen lisäksi ostopainikkeisiin lisätään toiminnot ostoa varten.

```
PurchasingManager.initiatePurchaseRequest("Paketin_avain");
```

Tämän jälkeen lisätään vastauksenkäsittelijän määrittely AndroidManifest.xml tiedostoon seuraavasti:

```
<receiver an-
droid:name="com.amazon.inapp.purchasing.ResponseReceiver" >
            <intent-filter>
                <action
                    an-
droid:name="com.amazon.inapp.purchasing.NOTIFY"
                    an-
droid:permission="com.amazon.inapp.purchasing.Permission.NOTIFY"
/>
            </intent-filter>
        </receiver>
```

/9/

6.3 Chrome Web Store

Verkkoselaimessa pelattavassa versiossa kauppapaikan lähestymistapa poikkesi hieman kahdesta muusta versiosta. Varsinaisen maksujärjestelmän upottaminen suoraan peli-ikkunan sisälle paljastui melko vaikeaksi, jolloin ongelmaan keksittiin kiertotie. Sivulle, jossa peli-ikkuna sijaitsee, lisättiin normaalit HTML-painikkeet, jotka veivät eri ostopakettien PayPal-sivuille (kuva 5.) Tätä varten luotiin PayPal tili, jonka jälkeen lisättiin eri paketit ”Payment buttons”- ominaisuuden avulla.

PayPalin ipn-ominaisuutta käytettiin maksutapahtuman selvitykseen. Jos käyttäjä ostaa paketin PayPalin kautta, PayPalin serveri lähettää pyynnön myyjän serverille. Tämä serveri luo sähköpostin, joka lähetetään takaisin ostajalle. Sähköpostissa on avain, jonka avulla ostaja voi lunastaa paketin pelin sisältä. Tätä varten luotiin ohjelmakoodi, joka tarkistaa pyynnön alkuperän, luo avaimen ja lähettää sen vielä lopulta ostajalle. (liite 10.4) Tämän jälkeen on mahdollista aloittaa testaaminen PayPal developer-palvelussa.

7 KAUPPAPAIKKOJEN TESTAUS

7.1 Google Play

Google Play-version testaus aloitetaan lisäämällä halutun testaajan Google-tunnukset kehittäjäportaalin asetusten kautta. Käyttäjätilin lisäämisen jälkeen voidaan aloittaa testaus. Varsinaisten myyntipakettien avainten testauksessa vaaditaan, että ohjelman apk-asennustiedosto on allekirjoitettu. Allekirjoittaminen vie aikaa ja siksi varsinainen testaus kannattaa aloittaa Googlen antamilla staattisilla testi-avaimilla, jotka eivät vaadi allekirjoitusta. Avaimia on neljä ja jokaista voidaan käyttää erilaisiin tilanteisiin. Alla olevasta taulukosta selviää eri avaimet ja niiden käyttötarkoitus.

Avain	Käyttötarkoitus
android.test.purchased	Onnistuneen ostotapahtuman testaus
android.test.canceled	Keskeytetyn ostotapahtuman testaus
android.test.refunded	Rahojen palautuksen testaus
android.test.item_unavailable	Tuote ei ole saatavilla tapahtuman testaus

Taulukko 3. Testiavaimet

Tässä työssä kaikki paketit sisälsivät pelin sisällä käytettävää virtuaalivaluuttaa, joten testiavainta käytettiin vain onnistuneen ostotapahtuman testaamiseen. Työssä testattiin ostoikkunan aukeamista normaalisti, ja että onnistuneen ostotapahtuman jälkeen käyttäjä palkittiin oikealla määrällä virtuaalivaluuttaa. Onnistuneiden testiavainten testaamisen jälkeen voitiin aloittaa testaaminen varsinaisilla avaimilla. Tätä varten ohjelman apk-tiedosto tulee allekirjoittaa ja kyseisen version tulee olla Google Playn kehittäjäportaalissa käytössä olevana versiona. Ohjelmasta tu-

lee testata, että myyntipakettien painikkeet avaavat oikean paketin myynti-ikkunan, ja että myyntitapahtuman onnistuessa käyttäjä palkitaan oikealla määrällä virtuaalivaluuttaa. /11/

7.2 Amazon Appstore

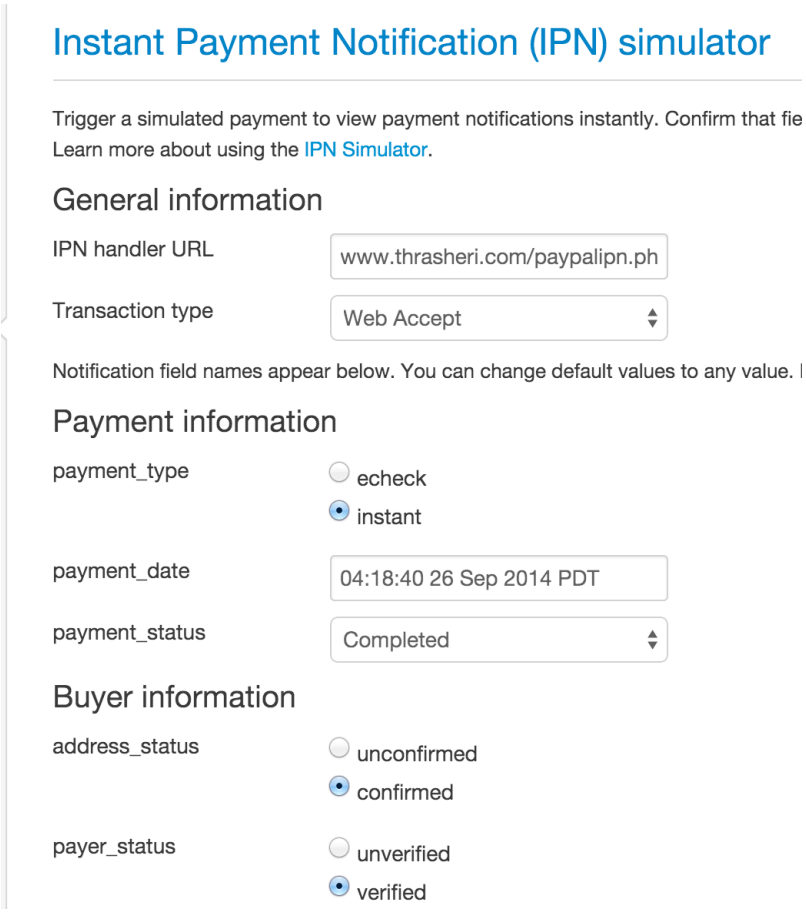
Amazon-version testaamista varten laitteeseen täytyy asentaa Amazonin luoma App tester-sovellus. Sovellus poimii testivaiheessa olevan ohjelman ostopyynnöt ja palauttaa sovelluksen asetuksista valitun vastauksen. Testausta varten tulee myös luoda JSON-kuvaus myyntipaketeista. Kuvauksen saa helpoiten lataamalla sen Amazonin kehittäjäportaalista myyntipakettien luomisen jälkeen. Latauksen jälkeen JSON-kuvaus tulee sijoittaa laitteen muistikortin juureen. Tämän jälkeen sovellus on valmiina testaukseen. Ohjelmasta testataan kaikkien pakettien toimivuus ja että ostotapahtuman onnistuessa ohjelma palkitsee käyttäjän oikealla määrällä virtuaalivaluuttaa. /12/

Kuvio 10. Amazon maksu-ikkuna

7.3 Chrome Web Store

Verkkoselaimessa pelattavan version maksujärjestelmänä toimi PayPal. PayPalin IPN-ominaisuutta on helpoin testata PayPalin developer-palvelussa, jossa pystyy testaamaan eri ominaisuuksia ilman laskutusta. Palvelun sisältä löytyy IPN-simulator testaussovellus, jota tässä työssä käytettiin. Sovellukseen syötetään ensin serverin osoite ja tyypiksi valitaan ”Web Accept”. Tämän jälkeen lisätään tuotteen nimi item_name kohtaan ja ostajan sähköpostiosoite payer_email kohtaan. Ostajan nimen ja muiden tietojen laittaminen on vapaaehtoista, sillä testaaminen onnistuu myös oletusasetuksilla.

Tietojen syöttämisen jälkeen painetaan ”send ipn”-painiketta, joka lähettää pyynnön kehittäjän serverille. Jos testaaminen on onnistunut testisähköpostiin saapuu viesti, jossa on avainkoodi valittuun tuotteeseen. /13/



Kuvio 11. PayPal IPN-simulator

8 AJATUKSIA JULKAISUN JÄLKEEN

Peli on ollut saatavilla eri alustoille jo noin 7 kuukautta. Tämän ajanjakson aikana peliä on ladattu yhteensä noin 35 tuhatta kertaa. Suurimman osan latauksista peli on saanut Google Play kauppapaikan kautta.

Alhaisen latausmäärän syynä on kova kilpailu ja samankaltaisten pelien suuri saatavuus. Google Play-kauppaan julkaistaan joka kuukausi yli 20 tuhatta uutta sovellusta. Suuren julkaisumäärän takia yhden sovelluksen on haastavaa erottua ryhmästä ilman kaupallista mainontaa.

Usealle alustalle julkaiseminen kannatti, koska kaikki versiot tuottavat rahaa. Pelimoottorin valintaan olisi kannattanut käyttää vielä enemmän aikaa. Pelimoottoriksi olisi ollut ehkä parempi valita jokin pelimoottori, jossa on tuki vielä useammalle alustalle. Kauppapaikkojen integrointi olisi helpottunut huomattavasti, jos pelimoottoriksi olisi valittu esimerkiksi Unity3D.

Julkaisua olisi voitu nopeuttaa tekemällä peliin vähemmän ratoja ja lisäämällä niitä myöhemmin, jos kysyntä sitä vaatii. Julkaisuhetkellä pelissä oli 240 rataa. Pelin olisi voinut julkaista esimerkiksi 90 radalla ja analytiikkatyökalun avulla seurata, kuinka moni pelaajista läpäisee kaikki radat.

Pelin tekemisestä ja julkaisusta jäi kaikin puolin hyvä mieli. Rahallisen hyödyn lisäksi peliä on hyvä käyttää referenssinä esimerkiksi töitä haettaessa. Lopullisessa versiossa on yli 16 tuhatta riviä ohjelmointikoodia.

9 YHTEENVETO

Opinnäytetyönä kehitetyn pelin sisälle luotiin toimivat maksujärjestelmät kolmelle eri kauppapaikalle. Kauppapaikkoina toimivat Google Play, Amazon Appstore ja Chrome webstore. Peli kehitettiin Libgdx-pelimoottoria hyödyntäen. Pelimoottoriin tutustumiseen ja sen ominaisuuksien oppimiseen kului paljon aikaa.

Pelin luomisen jälkeen oli vuoro suunnitella pelin sisällä toimiva kauppa. Google Play ja Amazon appstore toteutukset ovat käyttäjän näkökulmasta lähes identtiset. Mobiiliversioden sisällä on painike, joka vie kauppaan, jossa on neljä eri hintaista pakettia. Paketin oston jälkeen pelaaja palkitaan oikealla määrällä virtuaalivaluuttaa. Chrome Webstore-toteutuksessa käytettiin PayPal-maksujärjestelmää, jossa kauppapaikka oli peli-ikkunan vieressä, eikä upotettuna pelin sisälle. Paketin oston jälkeen käyttäjä saa viestin sähköpostiinsa, joka sisältää avaimen, jonka avulla on mahdollista lunastaa paketti pelin sisällä.

Eri kauppapaikkojen upottaminen Libgdx-pelimoottorilla tehtyyn peliin paljastui haasteelliseksi. Jokaiselle kauppapaikalle oli määrä tehdä oma toteutus ja kaikki toteutukset täytyi testata omalla tavallaan. Pelimoottorin valintaan olisi voinut käyttää vieläkin enemmän aikaa. Kauppapaikkojen integrointi olisi helpottunut huomattavasti, jos käytössä olisi ollut esimerkiksi Unity3D-pelimoottori, johon löytyy valmis liitännäinen kauppapaikkojen integrointiin.

Tämän jälkeen kaikki kolme maksujärjestelmää saatiin toimimaan halutulla tavalla ja toteutuksiin oltiin tyytyväisiä. Tulevaisuudessa maksujärjestelmien päivittäminen tulee olemaan työlästä, koska toteutuksia on kolme erilaista.

10 LÄHTEET

/1/ Mario Zechner, Robert Green 2012 Beginning Android Games 2-3

/2/ Patrick Niemeyer, Jonathan Knudsen 2005 Learning Java 1-5

/3/ Julie Meloni 2004 PHP 5 fast & easy web development xvi-xvii

/4/ SYBEX 2003 HTML COMPLETE 3-6

/5/ Eclipse sovelluksen kotisivut www.eclipse.org viitattu 14.5.2014

/6/ Libgdx pelimoottorin kotisivut http://libgdx.badlogicgames.com/ viitattu 14.5.2014

/7/ Soomla kirjaston kotisivut http://soom.la/ viitattu 14.5.2014

/8/ Soomla kirjaston dokumentointi

http://know.soom.la/docs/platforms/android/gettingstarted/ viitattu 14.5.2014

/9/ Amazon in app purchase dokumentointi

https://developer.amazon.com/public/apis/earn/in-app-purchasing/docs-v2/implementing-iap-2.0 viitattu 14.5.2014

/10/ PayPal IPN – dokumentaatio

https://developer.paypal.com/docs/classic/ipn/gs_IPN viitattu 14.5.2014

/11/ Android in app billing dokumentaatio viitattu 14.5.2014

http://developer.android.com/google/play/billing/billing_testing.html

/12/ Amazon in app billing testing dokumentaatio viitattu 14.5.2014

https://developer.amazon.com/public/apis/earn/in-app-purchasing/docs-v2/testing-iap-2.0

/13/ PayPal sandbox IPN-simulator viitattu 14.5.2014

https://developer.paypal.com/webapps/developer/applications/ipn_simulator

11 LIITTEET

11.1 Assets-luokka

```
import java.util.ArrayList;
import java.util.Arrays;

import com.soomla.store.IStoreAssets;
import com.soomla.store.domain.NonConsumableItem;
import com.soomla.store.domain.VirtualCategory;
import
com.soomla.store.domain.virtualCurrencies.VirtualCurrency;
import
com.soomla.store.domain.virtualCurrencies.VirtualCurrencyPac
k;
import com.soomla.store.domain.virtualGoods.VirtualGood;
import com.soomla.store.purchaseTypes.PurchaseWithMarket;

public class JewelsStoreAsset implements IStoreAssets {




    public static final String GEMS_ITEM_ID      = "g IN-
SERT_ID_HERE ";
    public static final String PILE_OF_GEMS_ID    = " IN-
SERT_ID_HERE ";
    public static final String BAG_OF_GEMS_ID = " IN-
SERT_ID_HERE ";
    public static final String SACK_OF_GEMS_ID   =  " IN-
SERT_ID_HERE ";
    public static final String CHEST_OF_GEMS_ID
="INSERT_ID_HERE";



    public static final VirtualCategory GENERAL_CATEGORY
= new VirtualCategory(
            "General", new Ar-
rayList<String>(Arrays.asList(new String[] {
```

```
PILE_OF_GEMS_ID, BAG_OF_GEMS_ID, SACK_OF_GEMS_ID,
CHEST_OF_GEMS_ID })));


    public static final VirtualCurrencyPack PILE_OF_GEMS
= new VirtualCurrencyPack(
            "Pile of gems",
// name
            "50 gems",
        // description
            PILE_OF_GEMS_ID,
// item id
            50,
// number of currencies in the pack
            GEMS_ITEM_ID,                                  //
the currency associated with this pack
            new PurchaseWithMarket(PILE_OF_GEMS_ID, 2.99));


    public static final VirtualCurrencyPack BAG_OF_GEMS =
new VirtualCurrencyPack(
            "Pile of gems",
// name
            "50 gems",
      // description
            BAG_OF_GEMS_ID,
// item id
            200,
// number of currencies in the pack
            GEMS_ITEM_ID,                                  //
the currency associated with this pack
            new PurchaseWithMarket(BAG_OF_GEMS_ID, 9.99));

    public static final VirtualCurrencyPack SACK_OF_GEMS
= new VirtualCurrencyPack(
            "Pile of gems",
// name
            "50 gems",
      // description
            SACK_OF_GEMS_ID,
// item id
            500,
// number of currencies in the pack
            GEMS_ITEM_ID,                                  //
the currency associated with this pack
```

```
            new PurchaseWithMarket(SACK_OF_GEMS_ID,
19.99));


    public static final VirtualCurrencyPack CHEST_OF_GEMS
= new VirtualCurrencyPack(
            "Pile of gems",
// name
            "50 gems",
       // description
            CHEST_OF_GEMS_ID,
// item id
            2000,
// number of currencies in the pack
            GEMS_ITEM_ID,                                  //
the currency associated with this pack
            new PurchaseWithMarket(CHEST_OF_GEMS_ID,
59.99));


    /** Virtual Currencies **/
    public static final VirtualCurrency GEMS = new Virtu-
alCurrency(
            "gems",
            "",
            GEMS_ITEM_ID
    );

     @Override
     public int getVersion() {
          // TODO Auto-generated method stub
          return 0;
     }

     @Override
     public VirtualCurrency[] getCurrencies() {
          return  new VirtualCurrency[] {
               GEMS
       };
     }

     @Override
     public VirtualGood[] getGoods() {
          // TODO Auto-generated method stub
          return new VirtualGood[]{
```

```
        };
    }

    @Override
    public VirtualCurrencyPack[] getCurrencyPacks() {
        return new VirtualCurrencyPack[] {
            PILE_OF_GEMS, BAG_OF_GEMS, SACK_OF_GEMS,
CHEST_OF_GEMS
        };
    }

    @Override
    public VirtualCategory[] getCategories() {
        // TODO Auto-generated method stub
        return new VirtualCategory[]{
            GENERAL_CATEGORY
        };
    }

    @Override
    public NonConsumableItem[] getNonConsumableItems() {
        // TODO Auto-generated method stub
        return new NonConsumableItem[]{

        };
    }


}
```

11.2 Eventhandler-luokka

```
import android.app.Activity;
import android.os.Handler;
import android.widget.Toast;

import com.soomla.store.BusProvider;
import com.soomla.store.SoomlaApp;
import com.soomla.store.StoreConfig;
import com.soomla.store.events.BillingNotSupportedEvent;
import com.soomla.store.events.BillingSupportedEvent;
import com.soomla.store.events.ClosingStoreEvent;
import com.soomla.store.events.CurrencyBalanceChangedEvent;
```

```
import com.soomla.store.events.GoodBalanceChangedEvent;
import com.soomla.store.events.GoodEquippedEvent;
import com.soomla.store.events.GoodUnEquippedEvent;
import com.soomla.store.events.ItemPurchaseStartedEvent;
import com.soomla.store.events.ItemPurchasedEvent;
import com.soomla.store.events.OpeningStoreEvent;
import com.soomla.store.events.PlayPurchaseCancelledEvent;
import com.soomla.store.events.PlayPurchaseEvent;
import com.soomla.store.events.PlayPurchaseStartedEvent;
import com.soomla.store.events.RestoreTransactionsEvent;
import
com.soomla.store.events.RestoreTransactionsStartedEvent;
import com.soomla.store.events.UnexpectedStoreErrorEvent;
import com.squareup.otto.Subscribe;


public class JewelsStoreEventHandler {

    private Handler mHandler;
    private Activity mActivityI;
    public JewelsStoreEventHandler(Handler handler, Activi-
ty activityI){
        mHandler = handler;
        mActivityI = activityI;

        BusProvider.getInstance().register(this);
    }

    @Subscribe
    public void onMarketPurchase(PlayPurchaseEvent mar-
ketPurchaseEvent) {




if(marketPurchaseEvent.getPurchasableVirtualItem().getItemI
d().equals(JewelsStoreAsset.PILE_OF_GEMS.getItemId())){

            Toast.makeText(mActivityI, "Thank you for buy-
ing pile of gems. The game is now ad-free.",
Toast.LENGTH_LONG).show();
//GIVE USER 50 GEMS HERE
}
```

```
if(marketPurchaseEvent.getPurchasableVirtualItem().getItem
Id().equals(JewelsStoreAsset.BAG_OF_GEMS.getItemId())){
            Toast.makeText(mActivityI, "Thank you for buy-
ing bag of gems. The game is now ad-free.",
Toast.LENGTH_LONG).show();
//GIVE USER 200 GEMS HERE
        }


if(marketPurchaseEvent.getPurchasableVirtualItem().getItemI
d().equals(JewelsStoreAsset.SACK_OF_GEMS.getItemId())){
            Toast.makeText(mActivityI, "Thank you for buy-
ing sack of gems. The game is now ad-free.",
Toast.LENGTH_LONG).show();
//GIVE USER 500 GEMS HERE
        }

if(marketPurchaseEvent.getPurchasableVirtualItem().getItemI
d().equals(JewelsStoreAsset.CHEST_OF_GEMS.getItemId())){
            Toast.makeText(mActivityI, "Thank you for buy-
ing chest of gems. The game is now ad-free.",
Toast.LENGTH_LONG).show();
//GIVE USER 2000 GEMS HERE
        }

    }

    @Subscribe
    public void onVirtualItemPurchased(ItemPurchasedEvent
itemPurchasedEvent) {
        show-
ToastIfDebug(itemPurchasedEvent.getPurchasableVirtualItem().
getName() + " was just purchased");
    }

    @Subscribe
    public void onVirtualGoodEquipped(GoodEquippedEvent
virtualGoodEquippedEvent) {
        show-
ToastIfDebug(virtualGoodEquippedEvent.getGood().getName() +
" was just equipped");
    }

    @Subscribe
    public void onVirtualGoodUne-
```

```
quipped(GoodUnEquippedEvent virtualGoodUnEquippedEvent) {
        show-
ToastIfDebug(virtualGoodUnEquippedEvent.getGood().getName()
+ " was just unequipped");
    }

    @Subscribe
    public void onBillingSupported(BillingSupportedEvent
billingSupportedEvent) {
        showToastIfDebug("Billing is supported");
    }

    @Subscribe
    public void onBillingNotSupport-
ed(BillingNotSupportedEvent billingNotSupportedEvent) {
        showToastIfDebug("Billing is not supported");
    }

    @Subscribe
    public void onPlayPurchas-
eStartedEvent(PlayPurchaseStartedEvent marketPurchas-
eStartedEvent) {
        showToastIfDebug("Market purchase started for: " +
marketPurchas-
eStartedEvent.getPurchasableVirtualItem().getName());
    }

    @Subscribe
    public void onPlayPurchase-
CancelledEvent(PlayPurchaseCancelledEvent marketPurchase-
CancelledEvent) {
        showToastIfDebug("Market purchase started for: " +
marketPurchase-
CancelledEvent.getPurchasableVirtualItem().getName());
    }

    @Subscribe
    public void onItemPurchas-
eStartedEvent(ItemPurchaseStartedEvent itemPurchas-
eStartedEvent) {
        showToastIfDebug("Item purchase started for: " +
itemPurchas-
eStartedEvent.getPurchasableVirtualItem().getName());
    }
```

```
    @Subscribe
    public void onClosingStore(ClosingStoreEvent clos-
ingStoreEvent) {
       // mActivityI.robotBackHome();

        showToastIfDebug("Going to close store");
    }

    @Subscribe
    public void onUnexpectedErrorIn-
Store(UnexpectedStoreErrorEvent unexpectedStoreErrorEvent) {
        showToastIfDebug("Unexpected error occurred !");
    }

    @Subscribe
    public void onOpeningStore(OpeningStoreEvent open-
ingStoreEvent) {
        showToastIfDebug("Store is opening");
    }

    @Subscribe
    public void onCurrencyBal-
anceChanged(CurrencyBalanceChangedEvent currencyBal-
anceChangedEvent) {
        showToastIfDebug("(currency) " + currencyBal-
anceChangedEvent.getCurrency().getName() + " balance was
changed to " + currencyBalanceChangedEvent.getBalance() +
".");
    }

    @Subscribe
    public void onGood-
BalanceChanged(GoodBalanceChangedEvent good-
BalanceChangedEvent) {
        showToastIfDebug("(good) " + good-
BalanceChangedEvent.getGood().getName() + " balance was
changed to " + goodBalanceChangedEvent.getBalance() + ".");
    }

    @Subscribe
    public void onRestoreTransactionsE-
vent(RestoreTransactionsEvent restoreTransactionsEvent) {
        showToastIfDebug("restoreTransactions: " + re-
storeTransactionsEvent.isSuccess() + ".");
    }
```

```
    @Subscribe
    public void onRestoreTransac-
tionsStartedEvent(RestoreTransactionsStartedEvent re-
storeTransactionsStartedEvent) {
        showToastIfDebug("restoreTransactions Started");
    }

    private void showToastIfDebug(final String msg) {
        if (StoreConfig.logDebug){
            mHandler.post(new Runnable() {
                @Override
                public void run() {
                    Toast toast =
Toast.makeText(SoomlaApp.getAppContext(), msg, 2000);
                    toast.show();
                }
            });
        }
    }
}
```

11.3 Amazon In App purchase observer

```
public class AmazonIAPobserver extends BasePurchas-
ingObserver {
     //   private static final String TAG = "IAPPurchas-
ingObserver";
      //  private boolean rvsProductionMode = false;
      //  private String currentUserID = null;

        public AmazonIAPobserver(Activity iapActivity) {
           super(iapActivity);
        }

        public void onSdkAvailable(final boolean is-
SandboxMode) {

          //   rvsProductionMode = !isSandboxMode;
        }
        public void onGetUserIdResponse(final GetUse-
rIdResponse response) {
             if (response.getUserIdRequestStatus() ==
                     GetUserIdRe-
```

```
sponse.GetUserIdRequestStatus.SUCCESSFUL) {
                    //     currentUserID = re-
sponse.getUserId();
                    }
                    else {
                        // Fail gracefully.
                    }
            }
            public void onItemDataResponse(final ItemDa-
taResponse response) {}
            public void onPurchaseResponse(final Pur-
chaseResponse response) {
                final PurchaseResponse.PurchaseRequestStatus
status= response.getPurchaseRequestStatus();

                if (status == Pur-
chaseResponse.PurchaseRequestStatus.SUCCESSFUL) {
                    Receipt receipt = response.getReceipt();
                    // ItemType itemType = re-
ceipt.getItemType();
                    String sku = receipt.getSku();
                    //  String purchaseToken = re-
ceipt.getPurchaseToken();

                    //Give user Ad-free version

                    // Store receipt and enable access to con-
sumable

                    if(sku.equals("pile_jewels_key")){
                        // Analyt-
icsTool.getPlatformResolver().saySomething("SKU: "+sku+"
ITEMTYPE: "+itemType.toString()+" PURCHASETOKEN
"+purchaseToken);
                        //Give 50 gems
                    }
                    else if(sku.equals("bag_jewels_key")){
                        //  Analyt-
icsTool.getPlatformResolver().saySomething("SKU: "+sku+"
ITEMTYPE: "+itemType.toString()+" PURCHASETOKEN
"+purchaseToken);
                        //Give 200 gems
                    }
                    else if(sku.equals("sack_jewels_key")){
                        //     Analyt-
```

```
icsTool.getPlatformResolver().saySomething("SKU: "+sku+"
ITEMTYPE: "+itemType.toString()+" PURCHASETOKEN
"+purchaseToken);
                        //Give 500 gems
                }
                else if(sku.equals("chest_jewels_key")){
                    //      Analyt-
icsTool.getPlatformResolver().saySomething("SKU: "+sku+"
ITEMTYPE: "+itemType.toString()+" PURCHASETOKEN
"+purchaseToken);

//Give 2000 gems            }
                else{

                }
            }


        }
        public void onPurchaseUpdatesResponse(final Pur-
chaseUpdatesResponse response) {}
    }
```

11.4 Paypal ipn

```
<?php



function getPackageType($type){

        switch($type){

                //Here the script creates the key

        }



}
```

```
function getChar($number)
{
        switch ($number)
                {
                //Here the script creates the key

                }
}
function solveGameAndPackage(){
$item_name      = $_POST['item_name'];

switch($item_name){

        //Here the script creates the key

}

}
function sendJewelsWorldsKey($packageType){
```

```
//Here the script creates the key



$item_name        = $_POST['item_name'];

$email = $_POST['payer_email'];

$gameUrl ="http://www.thrasheri.com/test/jewels.html";



$to      = $email;

$subject = "JEWELS WORLDS GEM-KEY";

$message = "


Thank you for your purchase


You purchased ".$item_name."


Your Gem-key is
```

```
------------------------

KEY: ".$key."

------------------------


You can use it inside the game.

Play the game at ".$gameUrl."


Thank you for your purchase!

";




mail($to, $subject, $message);


}


//read the post from PayPal system and add 'cmd'

$req = 'cmd=_notify-validate';


foreach ($_POST as $key => $value) {
```

```
    $value = urlencode(stripslashes($value));
    $req .= "&$key=$value";
}

//post back to PayPal system to validate
$header = "POST /cgi-bin/webscr HTTP/1.1\r\n";
$header .= "Content-Type: application/x-www-form-urlencoded\r\n";
$header .= "Host: www.paypal.com\r\n";

$header .= "Connection: close\r\n";
$header .= "Content-Length: " . strlen($req) . "\r\n\r\n";
$fp = fsockopen ('ssl://www.paypal.com', 443, $errno, $errstr, 30);

//

//error connecting to paypal
```

```
if (!$fp) {
    //
}

//successful connection
if ($fp) {
    fputs ($fp, $header . $req);

    while (!feof($fp)) {
        $res = fgets ($fp, 1024);
        $res = trim($res); //NEW & IMPORTANT

        if (strcmp($res, "VERIFIED") == 0) {

                            solveGameAndPackage();
            }

        if (strcmp ($res, "INVALID") == 0) {
                    //PURCHASE NOT VERIFIED DO NOT SEND MAIL
                            }
```

```
    }


    fclose($fp);
}


?>
```