MOBIILIPELIN OHJELMOINTI CORONA SDK:LLA

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, syksy 2013

Satu Minkkinen

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

VISAMÄKI
Tietojenkäsittelyn koulutusohjelma

Tekijä Satu Minkkinen **Vuosi** 2013

Työn nimi Mobiilipelin ohjelmointi Corona SDK:lla

TIIVISTELMÄ

Työn toimeksiantajana on Aatos Media, Hämeenlinnassa toimiva pieni yritys, joka suunnittelee, toteuttaa ja ylläpitää www-sivustoja, räätälöi julkaisujärjestelmien sivupohjia, konsultoi ja tekee hakukoneoptimointeja. Keväällä 2013 Aatos Media käynnisti mobiilipeliprojektin, jonka tavoitteena on saada tuotantoon toimiva, kiinnostava ja opettava esikouluikäisille suunnattu mobiilipeli, jolla tavoitellaan myös taloudellista hyötyä ja jatkuvuutta.

Opinnäytetyön tavoitteena oli saada valmiiksi 2–3 tasoinen, esimerkkigrafiikoilla toimiva tehtävä mobiilipeliin käyttämällä Corona SDK:ta. Opinnäytetyössä kerrotaan Corona SDK:sta, Lua-ohjelmointikielestä, Androidista sekä yleensä ottaen mobiilipeleistä ja niiden viimeaikaisesta nopeasta kehityksestä. Teoriaosuudessa käsitellään myös pelin suunnittelussa huomioitavia asioita mm. pelin idea ja tavoite, toteutus ja testaus.

Ohjelmoinnin osuus toteutettiin pääsääntöisesti internetistä löytyvien opetusvideoiden ja muiden ohjelmoijien laatimien tutoriaalien perusteella sekä kahden e-kirjan avulla. Valmiin pelin koodi toimitetaan toimeksiantajalle. Teoriaosuuden tietojen kerääminen tapahtui myös internetin kautta, pääsääntöisesti ohjelmistovalmistajien omien sivujen tiedoista.

Opinnäytetyön tuloksena voidaan esittää, että Corona SDK ja Lua-ohjelmointikieli ovat tarkoituksenmukaiset välineet mobiilipelien ja sovellusten ohjelmoimiseen. Ohjelmointikielen yksinkertainen rakenteen ja Corona SDK:n alustariippumattomuuden ansioista aloittelijakin voi saada aikaiseksi toimivan sovelluksen.

Avainsanat Android, Corona SDK, Lua, ohjelmointi**,** peliohjelmointi, sovelluskehittimet

Sivut 32 s.

Visamäki
Degree Programme in Business Information Technology

Author Satu Minkkinen **Year** 2013

Subject of Bachelor's thesis Programming a mobile game with Corona SDK

ABSTRACT

This thesis was commissioned by Aatos Media, a small company in Hämeenlinna, which plans, executes and maintains web-pages, customizes publishing systems page templates, consults and does search engine optimization. In the spring of 2013 Aatos Media launched a mobile game project, which aims to produce a functional, interesting and educational mobile game for preschoolers. Its purpose is to gain economic benefit and continuation.

The aim of this thesis was to complete a 2 to 3 level mobile game with simple graphics by using Corona SDK. The thesis describes Corona SDK, Lua-programming language, Android and mobile games in general and their recent rapid development. The theoretical part also discusses factors that are important to pay attention to, for example game idea and purpose, execution and testing.

The programming part was carried out mainly using teaching videos and tutorials made by other programmers found in the internet and two e-books. The game code of the finished game will be supplied to the commissioner. Internet was also used to collect information to the theoretical part, mainly by searching from software manufacturers' own web-pages.

As a result of this thesis it can be stated that Corona SDK and Lua programming language are appropriate tools for developing mobile games and software. The simple structure of the programming language and cross-platform support of the Corona SDK make it possible for beginners to create functional applications.

Keywords Android, Corona SDK, Lua, programming**,** game programming, software development.

Pages 32 p.

Sanasto

Android	Avoimeen lähdekoodiin perustuva käyttöjärjestelmä, joka on osa erilaisille mobiililaitteille tarkoitettua ohjelmistopinoa.
API, Application programming interface	Ohjelmointirajapinta, jonka avulla eri ohjelmat voivat keskustella keskenään.
Assosiatiivinen taulukko	Tieto tallennetaan taulukon soluun, johon indeksin sijasta viitataankin merkkijonolla.
BlackBerry	Kanadalaisen Research In Motionin, RIM:in, langaton mobiililaite.
Box2D	Avoimen lähdekoodin peleille tarkoitettu 2D-fysiikkamoottori, jota on käytetty muun muassa Angry Birdsissä.
BREW, Binary Runtime Environment for Wireless	Qualcomnin luoma kehitysalusta matkapuhelimille.
C++	Ohjelmointikieli
Garbage Collection, GC	Roskienkeräin, liittyy automaattiseen muistinhallintaan.
Git	Avoimeen lähdekoodiin perustuva, hajautettu versionhallintajärjestelmä.
GitHub	Lähdekoodin hallinta- ja jakopalvelu, joka hyödyntää Git-versionhallintaa.
Google Play	Googlen omistama sisältöpalvelu musiikin, elokuvien, kirjojen, lehtien ja Android-laitteiden sovelluksien hankintaan.
HTC Corporation	Taiwanilainen älypuhelinten valmistaja.
HTML5	Uusin versio HTML-merkintäkielestä, jota käytetään verkkosivujen tekemiseen.
IDC, International Data Corporation	Tutkimusyhtiö, joka seuraa muun muassa tietotekniikan ja tietoliikenteen markkinakehitystä.
IDE, Integrated development environment	Integroitu ohjelmistoympäristö, joka tarjoaa monipuolisia mahdollisuuksia ohjel-

	mistokehitykseen.
iOS	Applen kehittämä käyttöjärjestelmä iPhone-, iPod touch -, iPad- ja Apple TV -laitteisiin.
iTunes	Applen julkaisema soitin, joka kokoaa musiikit ja elokuvat yhteen paikkaan. Sisältää verkkomusiikkikaupan iTunes Storen sekä App Store -sovelluskaupan.
JSON, JavaScript Object Notation	Kevyt tiedonsiirtoformaatti, jota yleensä käytetään siirrettäessä tietoa palvelimen ja web-sovelluksen välillä.
Kindle Fire	Amazonin lanseeraama tablet-tietokone
Klusteri	Taloustieteessä yritysten ja yhteisöjen muodostamia maantieteellisiä keskittymiä, jotka ovat muodostuneet toisiinsa sidoksissa olevista toimialoista ja niihin liittyvistä muista toimijoista, jotka ovat merkittäviä kilpailun kannalta
MIT-lisenssi	Vapaa ohjelmistolisenssi, joka antaa käyttäjälle oikeuden muokata, kopioida ja käyttää teosta omassa projektissa sillä ehdolla, että lisenssin teksti säilyy lähdekoodissa.
Nook	Kirjakauppaketju Barnes & Noble:n lanseeraama lukulaite ja tablet-tietokone
Open Handset Alliance	Googlen johtama, teknologiayritysten muodostama liittouma, joka vastaa Androidin kehittämisestä.
OpenGL ES, OpenGL for Embedded Systems	Sulautetuille järjestelmille suunnattu laitteistoriippumaton ohjelmointirajapinta graafisia toimintoja varten.
OpenGL, Open Grphics Library	Laitteistoriippumaton ohjelmointirajapinta graafisia toimintoja varten.
Proseduraalinen ohjelmointi	Ohjelman suoritus jaetaan aliohjelmiin, proseduureihin, jota voidaan kutsua mistä tahansa pääohjelmasta tai muista aliohjelmista.

SDK, Software development kit	Sovelluskehitysalusta
Semantiikka	Ohjelmointikielen looginen merkitys
Skriptikieli eli komentosarjakieli	Kieli, jolla on helppo kirjoittaa skriptejä eli komentosarjoja, joiden avulla automatisoidaan tehtäviä
Syntaksi	Lauseoppi, joka muodostuu ohjelmointikielen sanastosta ja kielioppisäännöistä
Tarball	Tar-ohjelmalla tehty tiedostoarkisto, joka on yleensä pakattu.
WAP, Wireless Application Protocol	Protokolla, jonka avulla matkapuhelimilla voidaan selata internetiä langattomasti. Väistyvää tekniikkaa, Suomalaiset teleoperaattorit lakkauttavat wap-palveluita vähitellen.

KUVALUETTELO

TAULUKKOLUETTELO

SISÄLLYS

1 JOHDANTO

Opinnäytetyön toimeksiantajana on Aatos Media, jonka tavoitteena on saada tuotantoon toimiva, kiinnostava ja opettava esikouluikäisille suunnattu mobiilipeli, jolla tavoitellaan myös taloudellista hyötyä ja jatkuvuutta. Tavoitteen mahdollistava mobiilipeli-projekti on käynnistynyt jo ennen tämän opinnäytetyön aloittamista. Projektissa on mukana useampia henkilöitä, jotka osallistuvat pelin tekemiseen muun muassa laatimalla peliin tulevan grafiikan, musiikin, peliäänet, koodin ja huolehtimalla markkinoinnista.

Opinnäytetyön tavoitteena on suunnitella ja toteuttaa toimiva, 2–3 tasoinen, esimerkkigrafiikoilla toimiva tehtävä mobiilipeliin. Alun perin suunnittelussa oli tarkoitus kiinnittää erityistä huomiota kohderyhmän asettamiin rajoituksiin, esimerkiksi lukutaidottomuuteen, mutta toimeksiantoa muutettiin sen verran, että opinnäytetyössä keskitytään enemmän koodin luomiseen, ei niinkään grafiikkaan. Mobiilipeli toteutetaan käyttämällä Corona SDK -sovelluskehitysalustaa, joka antaa mahdollisuuden ohjelmoida pelejä niin, että ne sopivat useille eri mobiilialustoille. Tässä opinnäytetyössä keskitytään vain Androidiin. Opinnäytetyössä kerrotaan Corona SDK:sta, Lua-ohjelmointikielestä, Androidista, yleensä ottaen mobiilipeleistä sekä niiden viimeaikaisesta nopeasta kehityksestä. Teoriaosuudessa käsitellään myös pelin suunnittelussa huomioitavia asioita kuten pelin idea ja tavoite, toteutus ja testaus. Koska mobiilipelillä tavoitellaan myös taloudellista hyötyä ja jatkuvuutta, pohditaan millaiset ominaisuudet tekevät pelistä kiinnostavan ja koukuttavan.

Olennainen osa opinnäytetyötä on ohjelmakoodin tuottaminen. Opinnäytetyössä käsitellään seuraavia asioita: aloitusvalikon ja pelin ohjelmoiminen, koodin muokkaaminen niin, että saadaan 2–3 tasoa, pisteiden seuranta, onnistumisesta tai epäonnistumisesta ilmoittaminen, pelin lopettaminen ja tulosten tallentaminen. Lopuksi pelin toimivuutta testataan puhelimella. Opinnäytetyön yhteydessä tuotetut koodit eivät sisälly tähän raporttiin, vaan ne toimitetaan toimeksiantajalle.

2 CORONA SDK

Corona SDK, tekstissä tästä eteenpäin Corona, on Corona Labs Incin, entisen Ansca Mobilen, luoma laiteriippumaton sovelluskehitysalusta, jonka avulla voidaan yhdeltä koodipohjalta luoda erilaisia sovelluksia ja pelejä iOS:lle, Androidille, Nook:lle ja Kindle Fire:lle. Opinnäytetyötä kirjoitettaessa, mobiili.fi-sivuston 9.5.2013 julkaistun artikkelin mukaan, Nook Media on luopumassa Android-tableteistaan niiden huonon menestyksen vuoksi (Mobiili 2013). Coronan laitteistoriippumaton kirjasto ratkaisee miltei kaikki ongelmat erilaisten mobiilialustojen eroavuuksissa. Kirjasto toimii ohjelmointirajapintana, API, joka toimii lähestulkoon samalla tavalla kaikkien tukemiensa alustojen kanssa. Samalla kirjasto toimii abstraktina kerroksena, joka kätkee käyttäjältä alustojen väliset eroavaisuudet. (Coronalabs 2013.)

Corona käyttää Lua-ohjelmointikieltä, joka on kerrostettu C++ ja OpenGL:n päälle. Se käyttää OpenGL ES -grafiikkakirjastoa, joka tarjoaa funktioita ja toimintoja grafiikan näyttämiseen. Tämä mahdollistaa sen, että ohjelmoijan ei tarvitse juurikaan optimoida Coronalla tehtyjä pelejä tai sovelluksia, sillä niiden pitäisi toimia melko nopeasti useimmilla laitteilla. (Zammetti 2013.)

2.1 Historia

Coronan loivat Walter Luh ja Carlos Icaza, jotka olivat aiemmin työskennelleet Adobella. Coronan beta-versio julkaistiin kesäkuussa 2009 ja versio 1.0 julkaistiin joulukuussa 2009, jolloin se tuki vain yhtä alustaa: Applen iOS:ia, erityisesti iPhonea. Julkaisun yhteydessä kuitenkin kerrottiin jo suunnitelmista muiden alustojen suhteen. (Zammetti 2013.)

Huhtikuussa 2010 versio 2.0 julkaistiin ja tästä alkoi huima kehitys, sillä tämä oli ensimmäinen laiteriippumaton julkaisu, joka tuki myös iPadia ja Androidia. Myöhemmin samassa kuussa julkaistiin beta-versio Corona Game Edition, joka sisälsi fysiikkamoottorin ja muita erityisesti pelikehitykseen tarkoitettuja kehittyneitä ominaisuuksia. (Zammetti 2013.)

Tammikuussa 2011 Corona julkaistiin myös Windowsille, jolloin kehittäjät pystyivät rakentamaan sovelluksia PC:llä, sitä ennen vain unix-pohjaiset alustat pystyivät käyttämään Coronaa. Huhtikuussa 2011 mukaan tulivat myös NOOKin väritabletit. Elokuussa 2011 Corona Labs julkisti alustan, jonka yhteistyökumppanien ja pilvipohjaisten analyysien avulla kehittäjien on helpompi julkistaa ja markkinoida sovelluksiaan. (Zammetti 2013.)

2.2 Ominaisuudet

Coronan avulla voidaan luoda monipuolisia ja korkealaatuisia 2D-sovelluksia, kuten pelejä, e-kirjoja ja erilaisia liiketoimintasovelluksia. Coronan täysin HTML5-yhteensopiva näkymä yhdistettynä OpenGL-pohjaisiin grafiikoihin ja efekteihin mahdollistaa HTML5-mobiilisovellusten luomisen. Corona on nopea: API:n ansiosta muun muassa kuvan lisääminen, objektin animointi ja fysiikkamoottorin käyttö vaatii vain muutaman rivin koodia. Samalla käytettävissä on OpenGL-pohjaisen alustan standardi ominaisuuksia, kuten Facebook, Googlen kartat, Box2D-fysiikkamoottori ja in-app ostot. Coronan avulla voidaan luoda sovelluksia useimmille tärkeille alustoille ja laitteille. Koodi kirjoitetaan kerran ja käännetään iOS:lle, Androidille ja Kindle Fire:lle sopivaksi. Erilaisten laitekokojen ja resoluutioiden käsittely on helppoa: ensin määritellään sisällölle valittu tila pikseleinä ja kerrotaan mitä metodia käytetään alueen skaalaamisessa. Ohjelmoinnin tulos ja tehdyt muutokset koodissa ovat saman tien nähtävissä simulaattorissa ja terminaalissa. (Coronalabs 2013.)

Kuva 1 Esimerkkikuvat simulaattorin ja terminaalin näkymästä.

```
local myImage = display.newImage( "image.png" )
```

Kuva 2 Esimerkkikoodi kuvan lisäämisestä (Corona Docs 2013a).

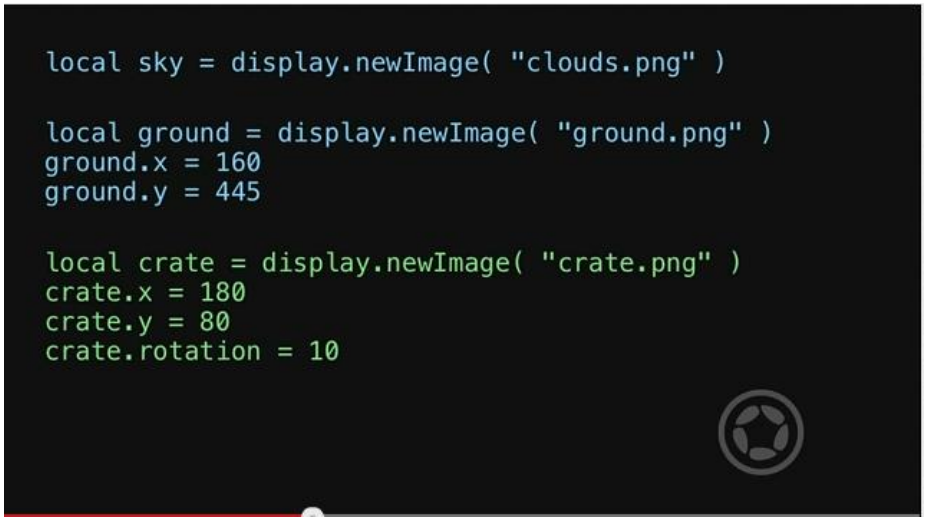
config.lua | main.lua | menuScene.lua | gameSce

```
1  application = {
2      content = {
3          width = 480,
4          height = 800,
5          scale = "letterBox",
6          fps = 30,
7
```

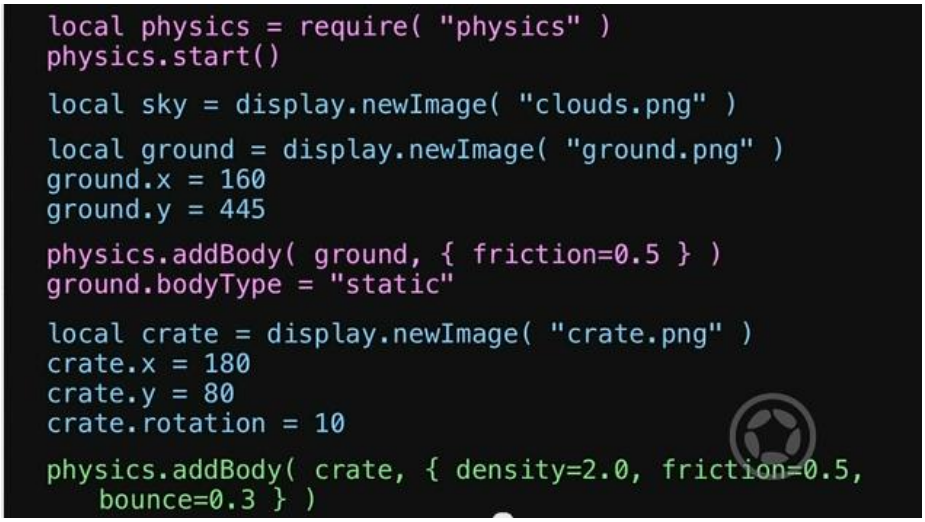
Kuva 3 Esimerkkikoodi resoluutioiden käsittelystä. (Corona Docs 2013b.)

Sisältöalue määritellään config.lua-tiedostossa leveytenä ja korkeutena. Jos jompikumpi näistä arvoista on 0, dynaamista skaalausta ei tehdä. Skaalausmetodeja on kolme: letterBox, zoomEven ja zoomStretch. LetterBox skaalaa sisältöalueen säilyttäen mittasuhteet ja estämällä sisältöä siirtymästä näytön ulkopuolelle. Tämä saattaa joidenkin laitteiden kohdalla tehdä näytön reunoille musta palkit. ZoomEven säilyttää mittasuhteet, mutta osa sisällöstä saattaa jäädä näytön reunojen ulkopuolelle. ZoomStretch skaalaa sisällön koko näytön kokoiseksi, mutta sisältö voi venyä vaaka- tai pystysuunnassa ja siitä johtuen siis vääristyä. Tämä on oletusmetodi joka on käytössä, jos skaalausparametri jätetään koodista pois.

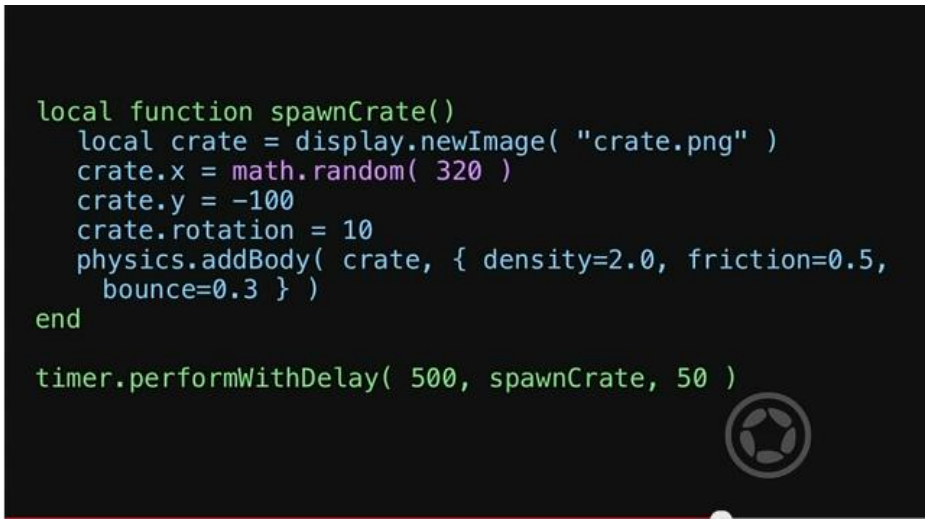
Seuraavissa kuvissa näytetään esimerkkikoodi fysiikkamoottorin käytöstä. (Youtube 2013.)



Kuva 4 Koodi pilvien, maan ja hahmon lisäämiseksi



Kuva 5 Koodi, jolla hahmo muutetaan liikkuvaksi hahmoksi.



Kuva 6 Koodi, joka kasvattaa hahmojen määrää

2.3 Coronan eri versiot

Coronasta löytyy kolme eri vaihtoehtoa: Corona SDK Starter, Corona SDK Pro ja Corona Enterprise. Corona SDK Starterilla käyttäjä voi luoda ja julkaista sovelluksiaan maksamatta mitään Corona Labs:lle. Corona Labs ei myöskään vaadi minkäänlaista osuutta mahdollisista tuloista. Corona Labs ei kuitenkaan julkaise käyttäjien tekemiä sovelluksia, vaan ne täytyy julkaista eri laitevalmistajien sovelluskaupoissa, kuten App Store tai Google Play, käyttämällä niiden omia kehitystyökaluja. Applen sovelluskaupassa kehittäjän pitää luoda kehittäjätunnus, joka maksaa 99 $ vuodessa (Mac Developer Program 2013). Google Play -kehittäjä taas maksaa kertaluonteisen 25 $ rekisteröintimaksun (Google Play 2013). Mikäli Coronan ilmaisesta versiosta ei löydy tarpeeksi ominaisuuksia, käyttäjällä on mahdollisuus päivittää ohjelma maksullisiin versioihin. Opinnäytetyötä kirjoitettaessa Corona SDK Pron hinta oli 599 $/vuosi ja Corona Enterprisen joko 999 $/vuosi tai 2499 $/vuosi riippuen ohjelman ominaisuuksista. (Coronalabs 2013a.)

2.4 Kehittäjäyhteisö

Coronan yhteisön avulla käyttäjän on kätevä kehittää taitojaan ja pysytellä viimeisimpien muutosten ja uudistusten mukana. Foorumeilla kehittäjät tarjoavat vinkkejä ja apua ongelmatilanteisiin sekä jakavat omia koodejaan. Coronan käyttäjäryhmät kokoontuvat eri puolilla maailmaa erilaisissa tapahtumissa ja sitä kautta pääsee oppimaan ja luomaan hyödyllisiä kontakteja. Yhteisön kehittäjät ovat luoneet Coronan ympärille hyödyllisiä työkaluja kuten Photoshop-pohjaisen työkalun e-kirjojen luomiseen ja Corona Project Managerin projektinhallintaan. (Coronalabs 2013a.)

3 LUA-OHJELMOINTIKIELI

Lua on nopea, kevyt, tehokas ja kooltaan pieni upotettava skriptikieli, joka on suunniteltu ja toteutettu vuonna 1993 Brasilialaisessa yliopistossa PUC-Rio:ssa, the Pontifical Catholic University of Rio de Janeiro. Kehittäjäryhmään kuuluivat Roberto Lerusalimschy, Waldemar Celes ja Luiz Henrique de Figueiredo, jotka toimivat edelleen ylläpitäjinä ja konsultteina. Lua on sertifioitu avoimen lähdekoodin sovellus, jota toimitetaan MIT-lisenssin ehtojen mukaisesti. Se on vapaasti käytettävissä, myös kaupallisissa tarkoituksissa, ilman minkäänlaisia kustannuksia. (Lua 2013a; Lua 2013b.)

Lua yhdistää yksinkertaisia proseduraalisia syntakseja tehokkaisiin assosiatiivisiin taulukoihin perustuviin datakuvausrakenteisiin ja laajennettavaan semantiikkaan. Lua toimii Unix ja Windows -ympäristössä sekä erilaisissa mobiililaitteissa. Luaa käytetään useissa sovelluksissa kuten Adobe Photoshop Lightroomissa, Apache HTTP serverissä, Firefoxissa sekä MediaWikissä. Lua on tällä hetkellä yksi suosituimmista skriptikielistä peliohjelmoinnissa, sitä käytetään muun muassa World of Warcraftissa ja Angry Birdsissä. Luan avulla World of Warcraftin pelaajilla on mahdollisuus muokata pelin käyttöliittymää oman mielensä mukaiseksi. Lua sopii hyvin konfigurointiin, skriptaamiseen ja protojen luomiseen, sillä se tyypitetään dynaamisesti, jolloin muuttujien tyyppiä ei tarvitse määritellä. Luan koodi käännetään omalle virtuaalikoneelle. Tämän lisäksi sillä on automaattinen muistinhallinta kasvavalla roskienkeräimellä, joka toimii viiveettä, hukkaamatta muistia eikä sen täytäntöönpano ole kovinkaan mutkikasta. Manuaalinen muistinhallinta on työlästä ja virhealtista ja tästä syystä automaattinen muistinhallinta toimii yleensä paremmin myös niissä laitteissa, joissa on muistiltaan rajalliset resurssit. (Lua 2013a; Lua 2013c; Lua 2013d.)

Koska Lua on kooltaan pieni, se on helppo upottaa sovelluksiin kasvattamatta liikaa sovelluksen kokoa. Tämä on erityisen tärkeää mobiililaitteiden rajoitetun muistin kanssa. Lua on toteutettu C-kielellä ja se sisältää kokonaisuudessaan noin 20 000 riviä koodia. Esimerkiksi Lua 5.2.2:n tarball-tiedosto, joka sisältää lähdekoodin ja dokumentaation, on pakattuna 246 kilotavua ja pakkaamattomana 960 kilotavua. Unixissa Luan tulkki vie tilaa 182 kilotavua ja kirjasto 243 kilotavua. Luan ydin ei käytä lainkaan staattista dataa. C:n API tarvitsee vain yhden käsittelijän, joka toimii koko virtuaalikoneessa. Luan ydin ei hukkaa muistia, vaan se voi hyödyntää yleisiä muistin kohdentamiseen liittyviä toimintoja. Jos tarve vaatii, Lua voi käyttää useita itsenäisiä tulkkeja samassa prosessissa. Lua tarvitsee vain muutamia C-kirjaston toimintoja toimiakseen ja näitäkin voidaan myöhemmin vähentää. Lua:lla on yksinkertainen ja hyvin dokumentoitu API, joka mahdollistaa integraation koodiin, joka on kirjoitettu toisella kielellä. Tästä syystä Luaa on helppo laajentaa muiden kielien kirjastoilla ja Luan avulla taas voidaan laajentaa muilla kielillä, kuten C, C++, C#, Java, Perl, kirjoitettuja ohjelmia. (Lua 2013a; Lua 2013d.)

Lua.org-sivuston mukaan Lua on useiden riippumattomien testien mukaan osoittautunut useissa vertailuarvoissa tulkattavien skriptikielien nopeim-

maksi kieleksi. Luan tulkki on viritetty laajoihin suorituksiin ja C:n kutsumekanismi on yksi nopeimmista. Lisää nopeutta vaativille Luaan on saatavilla myös itsenäinen toteutus, LuaJIT, joka käyttää ajonaikaista kääntäjää muuttaakseen koodin konekieleksi, mahdollistaen vielä tehokkaamman suorituksen vähemmillä resursseilla. Tämä on erityisen tärkeä ominaisuus puhelimia ohjelmoidessa, sillä niissä on yleensä vähän muistia ja teholtaan pienet prosessorit. (Lua 2013a; Lua 2013d.)

Lua on pelkistetty. Sen syntaksi mahtuu yhdelle sivulle ja sen semantiikka on johdonmukaista ja intuitiivista. Aloittelijatkin voivat alkaa ohjelmoimaan helposti, ohjeistusta on helposti löydettävissä, mitään IDE:ä ja SDK:ta ei tarvita ja mikä tahansa tekstieditori riittää. Lua:ssa on kehittyneitä lisätoiminoja kuten vuoroittaisrutiineja ja meta-mekanismeja. Luan suunnittelussa perusajatuksena on tarjota meta-mekanismeja ominaisuuksien toteuttamiseksi sen sijaan, että tarjottaisiin joukko ominaisuuksia suoraan ohjelmointikielellä. Tämä näkyy käyttäjälle esimerkiksi siten, että vaikka Lua ei ole puhdas olio-ohjelmointikieli, se tarjoaa keinoja luokkien ja periytyvyyden toteuttamisekseen. Meta-mekanismit vähentävät käsitteitä ja pitävät kielen pienenä, sallien samalla semantiikan laajennuksen omaperäisellä tavalla. Luan vuoroittaisrutiinit tarjoavat nopean ja muistiystävällisen tavan suorittaa multiajoja. Luan vuoroittaisrutiinit ovat sisäänrakennettuja ja käyttöjärjestelmäriippumattomia. Multiajolla mahdollistetaan esimerkiksi puhelinkeskustelun aikana tapahtuva puhelinnumeron tarkistus tai kalenterin käyttö. (Lua 2013a; Lua 2013d.)

4 VERSIONHALLINTA

Ohjelmiston versionhallinta mahdollistaa ohjelmiston hallitun kehityksen ja kehityksen seuraamisen. Versionhallinnan avulla on mahdollista räätälöidä ohjelmistoa erilaisille alustoille ja asiakkaille sopivaksi. Tarvittaessa versionhallinta antaa mahdollisuuden palata aiempaan kehitysversioon, jos uudemmassa versiossa on esim. toimintaan liittyviä ongelmia.

Versionhallinta muodostuu neljästä päätoimenpiteestä, joita ovat versiointi, versioiden merkitseminen, versioiden välisten erojen tunnistaminen ja versioiden tallentaminen. Versiointi voidaan jakaa historialliseen, loogiseen ja yhteistoiminnalliseen versiointiin. Historiallisessa versioinnissa uusi versio eli revisio, syrjäyttää aikaisemman version. Loogisessa versiossa luodaan haaroja, branch, joista jokainen edustaa uutta kehitysvaihetta. Yhteistoiminnallisessa versioinnissa luodaan väliaikaisia versioita, jotka myöhemmin yhdistetään toisiin versioihin. Historiallisessa versioinnissa revisiot merkataan yleensä numeerisesti siten, että suurin luku tarkoittaa viimeisintä revisiota. Loogisen ja yhteistoiminnallisen versioinnin yhteydessä merkitsemiseen käytetään useimmiten nimiä. Versionhallinnan muihin toimenpiteisiin nähden, versioiden välisten erojen tunnistaminen ei ole välttämätöntä, vaan päätavoitteena on säästää tallennustilaa. Erojen tunnistamiseen käytetään yleensä merkkivertailua tiedostojen välillä, mutta tämä toimii vain tekstitiedostoja käsiteltäessä. Tallennuspaikkana käytetään tietovarastoa, repository, jonne voidaan tallentaa yksittäinen tiedosto, hakemisto tai koko tietokanta. Versionhallinta voi olla keskitetty, hajautettu tai paikallinen. Keskitetyssä versionhallinnassa tietovarasto sijaitsee jollain

palvelimella ja varastoa voi käyttää useampi henkilö, joille on sallittu pääsy tietovarastoon. Hajautetussa versionhallinnassa tietovarasto voi olla julkinen, josta käyttäjät saavat käyttöönsä täydellisen kopion versiohistoriasta. Paikallisessa versionhallinnassa tietovarasto sijaitsee paikallisella tietokoneella. (linux.ictlab.kyamk 2006; Luong 2013; Reaktor 2013.)

4.1 Git-versionhallinta ja GitHub-lähdekoodinjakopalvelu

Git on Linus Torvaldsin kehittämä, avoimeen lähdekoodiin perustuva hajautettu versionhallintajärjestelmä, joka suunniteltiin Linux-ytimen kehitystiimin tarpeita vastaavaksi. Koska Linux-ytimeen tulee paljon muutoksia, versionhallinnan täytyy olla mahdollisimman nopea ja sen täytyy tukea hajautettua työskentelyä ja estää datan virheellisyyttä ja katoamista. Unixin periaatteiden mukaisesti Git ei ole yksittäinen sovellusohjelma, vaan joukko pienempiä ohjelmia, joista jokainen suorittaa omia toimintojaan. (Git 2013a; Git 2013b.)

GitHub on lähdekoodin hallinta- ja jakopalvelu, joka hyödyntää Git-versionhallintaa. Kuka tahansa voi rekisteröityä palveluun ja luoda sinne lähdekoodivarastoja. Ilmaisella käyttäjätilillä voidaan luoda julkisia varastoja, maksullisella käyttäjätilillä onnistuu myös yksityisten varastojen luonti. GitHubin graafisen käyttöliittymän avulla voidaan tarkastella eri lähdekooditiedostojen sisältöä sekä seurata niiden muutoksia versioin kehittyessä. Avoimelle lähdekoodille ominaista on jakaminen ja yhteistyö ja siksi kuka tahansa käyttäjä voi tehdä itselleen kopion julkisesta varastosta, lähteä kehittämään koodia oman mielensä mukaisesti ja myöhemmin tarjota muutoksia alkuperäiselle kehittäjälle. (Linux-Aktivaattori 2013.)

5 ANDROID

Android on avoimeen lähdekoodiin perustuva käyttöjärjestelmä, joka on osa erilaisille mobiililaitteille tarkoitettua ohjelmistopinoa. Android järjestelmät nimetään yleensä jonkin makean herkun mukaan, tällä hetkellä käytössä on Jelly Bean, ja uusin käyttöjärjestelmä Kitkat julkaistiin lokakuussa 2013. (Android 2013a; Android 2013b.)

5.1 Historia

Androidia hallinnoi joukko yrityksiä, joka tunnetaan nimellä Open Handset Alliance. Tällä hetkellä joukkoon kuuluu 84 teknologia- ja mobiilialan yritystä, joita johtaa Google. Lokakuussa 2003 Andy Rubin, Rich Miner, Nick Sears ja Chris White perustavat Android Inc:in. Yrityksen tarkoituksena oli rakentaa ohjelmistoa puhelimille ja digikameroille. Google osti Androidin vuonna 2005, Andy Rubinin jatkaessa tiiminsä kanssa Android-käyttöjärjestelmän rakentamista mobiililaitteille. Marraskuussa 2007 muodostettiin Open Handset Alliance ja julkistettiin Android Beta SDK. Vuonna 2008 Android aloitti yhteistyön T-Mobilen kanssa julkaistakseen ensimmäisen Android-älypuhelimen – G1:den. (Android 2013c, Business Insider 2013.)

Vuonna 2009 julkistettiin muun muassa käyttöjärjestelmät Cupcake, Donut ja Eclair. Samana vuonna alkoi Androidin todellinen nousu, kun marraskuussa julkaistiin Motorola Droid. Puhelin käytti uutta käyttöjärjestelmää, versiota 2.0, joka sisälsi myös Verizonin. Verizon Communications on yhdysvaltalainen tietoliikenneyritys, joka omistaa Verizon Wireless -matkapuhelinoperaattorin. Tämä oli suuri voitto, sillä tuolloin Verizon ei tukenut iPhonea. Motorola Droid aloitti markkinointitrendin, jolla Androidin valmistajat halusivat erottua iPhonesta. Motorola Droidin fyysistä näppäimistöä ja vaihdettavaa akkua mainostettiin kahtena iPhonen päihittävänä myyntivalttina. (Business Insider 2013.)

Google julkaisi ensimmäisen oman älypuhelimensa Nexus One:n tammikuussa 2010. Googlen tarkoitus oli muuttaa kuluttajien tapaa ostaa älypuhelimia. Google myi puhelinta vain verkossa hintaan 529 $. T-Mobilen asiakkaat saivat sen hankittua sopimuksella hintaan 179 $. Valitettavasti Nexus One osoittautui suutariksi ja Google lopetti sen tuotannon heinäkuussa 2010. Tästä huolimatta tämä oli hyvä vuosi Androidille. Tuolloin Samsung julkaisi ensimmäisen Galaxy S -älypuhelimen ja sen eri versioita. Puhelimesta tulee lopulta Androidin suosituin merkki: Samsung on tähän mennessä myynyt yli 100 miljoonaa Galaxy-puhelinta. Kesäkuussa 2010 HTC julkaisi EVO 4G:n, joka oli yksi parhaiten arvostettu älypuhelin ja oli monen mielestä paras vaihtoehto iPhonen sijaan. 2010 oli myös vuosi, jolloin Androidin markkinaosuus ensimmäistä kertaa ohitti iPhonen markkinaosuuden Yhdysvalloissa. Siitä huolimatta BlackBerry oli vielä huipulla. Loppuvuodesta 2010 Google antoi Nexus ohjelmalle uuden mahdollisuuden julkaisemalla Nexus S -puhelimen, jonka se oli tehnyt Samsungin kanssa. Nexus S oli rakenteeltaan itse asiassa miltei täsmälleen kuin Galaxy S, mutta muotoilua oli hieman muutettu. Se oli ensimmäinen puhelin, jossa oli Gingerbread: käyttöjärjestelmänä, josta tuli pian Androidin suosituin. (Business Insider 2013.)

Vuoden 2011 alussa Google yritti tehdä tableteille tarkoitetun Honeycomb version. Käyttöjärjestelmä julkaistiin Motorolan Xoom-tabletilla. Xoom ja muut Honeycomb-tabletit eivät menestyneet hyvin. Android ohitti BlackBerryn Yhdysvaltojen markkinoilla keväällä 2011. Siitä lähtien BlackBerryn osuus on ollut laskussa. Suurin muutos Androidille tapahtui marraskuussa 2011, kun uusi versio nimeltään Ice Cream Sandwich julkaistiin. Tämä käyttöjärjestelmä oli tarkoitettu toimimaan myös tableteilla ja se antoi uuden ilmeen Androidille. Samassa kuussa Samsung käynnisti massiivisen markkinointikampanjan Applea ja iPhonea vastaan. Siitä lähtien Android on käyttänyt kymmeniä miljardeja markkinointiin ja on tullut maailman suurimmaksi älypuhelinten valmistajaksi. Samsungin älypuhelimet ovat suosituimpia. Vuonna 2012 Samsung julkaisi Galaxy S III -älypuhelimen, josta tuli myydyin puhelin iPhonen jälkeen. (Business Insider 2013.)

Heinäkuussa 2012 julkaistaan Android 4.1 Jelly Bean. Marraskuuhun 2013 mennessä Andoidin 4.x-pääversio Jelly Bean löytyy yli 50 prosentista kaikista markkinoilla olevista Android-laitteista. Androidin uusin käyttöjärjestelmä KitKat julkaistiin 31.10.3013. (Android 2013a; visual 2013.)

5.2 Ominaisuudet

Google omistaa Androidin, joten Googlen omat sovellukset, kuten kartat, Google+ ja Drive toimivat hyvin Androidin kanssa. Markkinoilla on tällä hetkellä monia erilaisia Android-laitteita, lähinnä älypuhelimia ja tablettietokoneita. Laitteita löytyy erilaisilla ominaisuuksilla varustettuna, erihintaisia ja erikokoisia. Tunnettuja valmistajia ovat ainakin Samsung, HTC, LG, Motorola, Sony ja Asus. (Android 2013b.)

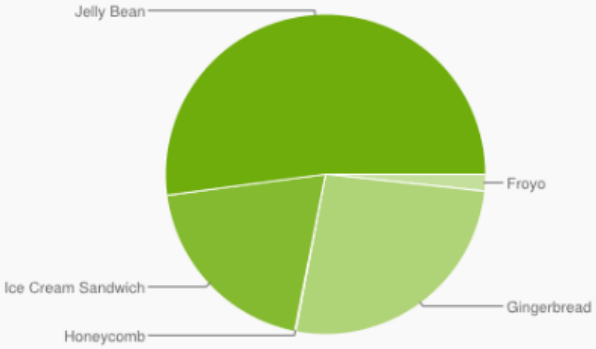
Monet yritykset ovat investoineet ja edelleen investoivat voimakkaasti Androidiin parantaakseen sitä ja tuodakseen lisää Android-laitteita markkinoille. Android perustuu tarkoituksella avoimeen lähdekoodiin. Joukko organisaatioita yhteisine tarpeineen on yhdistänyt resurssejaan luodakseen tuotteen, jota jokainen voi räätälöidä ja muokata omia tarpeitaan vastaavaksi. Hallitsematon muokkaaminen voi johtaa yhteensopimattomiin muutoksiin ja tämän estämiseksi käytettävissä on Android yhteensopivuus -ohjelma, Android Compatibility Program. Ohjelmassa kerrotaan mitä kehittäjiltä vaaditaan, jotta he saavat tehtyä Android-yhteensopivia sovelluksia. Kuka tahansa voi käyttää Androidin lähdekoodia, mutta mikäli aikoo rakentaa Androidiin sopivia sovelluksia, täytyy liittyä mukaan Android yhteensopivuus -ohjelmaan. (Android 2013c.)

5.3 Tilastoja Google Playn mukaan

Google Play -sovelluskauppa tilastoi asiakkaiden asennusten ja ostojen perusteella käyttöjärjestelmiä, laitteiden näyttöjen kokoa sekä tarkkuutta. Syyskuun 2013 jälkeen näissä tilastoissa ei ole ollut mukana laitteita joiden käyttöjärjestelmä on vanhempi kuin Android 2.2, sillä nuo laitteet eivät tue Google Play -sovelluskauppaa. (Android 2013d.)

Marraskuun 1. päivänä tilastojen mukaan Jelly Bean oli suosituin käyttöjärjestelmä 52,1 prosentin osuudella ja normaalikokoinen näyttö hdpi-tarkkuudella oli suosituin.

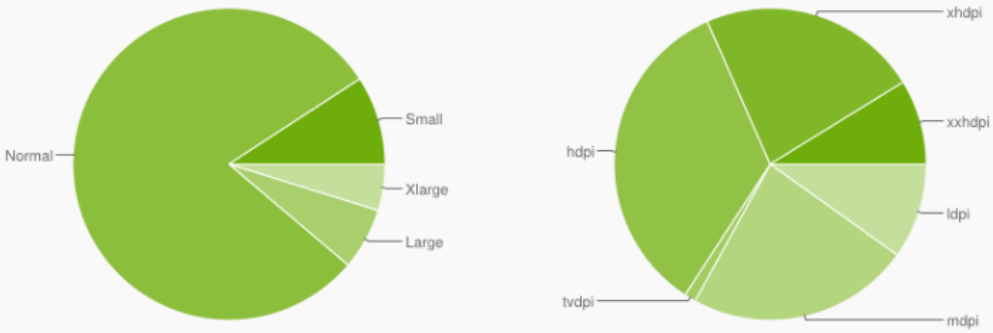
Version	Codename	API	Distribution
2.2	Froyo	8	1.7%
2.3.3 - 2.3.7	Gingerbread	10	26.3%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	19.8%
4.1.x	Jelly Bean	16	37.3%
4.2.x		17	12.5%
4.3		18	2.3%



Data collected during a 7-day period ending on November 1, 2013.
Any versions with less than 0.1% distribution are not shown.

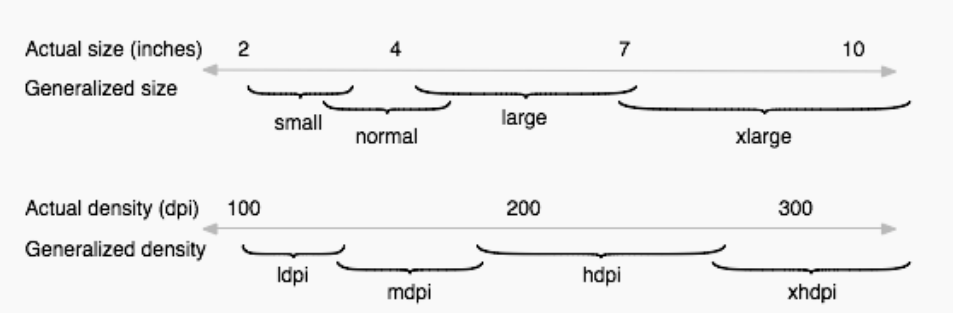
Kuva 8 Käyttöjärjestelmien osuus Google Playn mukaan (Android 2013e).

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	9.2%						9.2%
Normal	0.1%	15.1%		33.4%	22.2%	8.8%	79.6%
Large	0.6%	3.6%	1.2%	0.5%	0.5%		6.4%
Xlarge		4.4%		0.3%	0.1%		4.8%
Total	9.9%	23.1%	1.2%	34.2%	22.8%	8.8%	



Data collected during a 7-day period ending on November 1, 2013
Any screen configurations with less than 0.1% distribution are not shown.

Kuva 7 Näyttöjen koot ja tarkkuudet Google Playn mukaan (Android 2013e).



Kuva 9 Näyttöjen koot tuumina ja resoluutio (Android 2013e).

Android-laitteiden näytöt jaetaan erilaisiin tarkkuusluokkiin, jotka ovat 120, 160, 240, ja 320 pistettä tuumalle. Nimeltään tarkkuudet ovat ldpi, mdpi, hdpi ja xhdpi. Käyttöliittymää, peliä tai muuta sovellusta suunnitellessa kannattaisi huomioida eri laitteita. Kuvista voidaan tehdä omat versiot eri tarkkuuksille, ja ohjelma osaisi valita oikealla tarkkuudella olevan kuvan. Näin vähennetään skaalauksen aiheuttamaa mahdollista kuvan vääristymää.

Density	`ldpi`	Resources for low-density (*ldpi*) screens (~120dpi).
	`mdpi`	Resources for medium-density (*mdpi*) screens (~160dpi). (This is the baseline density.)
	`hdpi`	Resources for high-density (*hdpi*) screens (~240dpi).
	`xhdpi`	Resources for extra high-density (*xhdpi*) screens (~320dpi).
	`nodpi`	Resources for all densities. These are density-independent resources. The system does not scale resources tagged with this qualifier, regardless of the current screen's density.
	`tvdpi`	Resources for screens somewhere between mdpi and hdpi; approximately 213dpi. This is not considered a "primary" density group. It is mostly intended for televisions and most apps shouldn't need it—providing mdpi and hdpi resources is sufficient for most apps and the system will scale them as appropriate. If you find it necessary to provide tvdpi resources, you should size them at a factor of 1.33*mdpi. For example, a 100px x 100px image for mdpi screens should be 133px x 133px for tvdpi.

Kuva 10 Android-laitteiden tarkkuudet (Android 2013e).

5.4 Androidin kehitys markkinoilla

Android-sivuston mukaan käyttöjärjestelmä on käytössä sadoissa miljoonissa älypuhelimissa ja tableteissa yli 190 maassa. Se on laajimmin käytetty mobiilialusta ja sen osuus kasvaa koko ajan – päivittäin aktivoidaan yli miljoona uutta laitetta. (Android 2013f.)

Yhdysvaltain markkinoilla Androidin markkinaosuus on 52 %, kun taas maailmanlaajuisesti yli 80 % markkinoista on Androidin hallussa. Yhdysvalloissa tilanteeseen vaikuttaa se, että kytkykauppojen ansiosta iPhone ei aina ole kallein vaihtoehto. Tilanne markkinoilla saattaa muuttua, mikäli Apple julkaisee halvemman iPhonen. Tässä vaiheessa on jo nähtävissä, että Android valmistajat valmistautuvat vastaiskuun tuomalla markkinoille lisää edullisen kategorian puhelimia. (Puhelinvertailu 2013.)

Taulukko 1 IDC-sivuston tilastotietoa matkapuhelinten käyttöjärjestelmistä, määrät miljoonissa

Operating System	2Q13 Unit Shipments	2Q13 Market Share	2Q12 Unit Shipments	2Q12 Market Share	Year-over-Year Change
Android	187.4	79.3%	108	69.1%	73.5 %
iOS	31.2	13.2 %	26	16.6 %	20.0 %
Windows Phone	8.7	3.7 %	4.9	3.1 %	77.6 %
BlackBerry OS	6.8	2.9 %	7.7	4.9 %	-11.7 %
Linux	1.8	0.8 %	2.8	1.8 %	-35.7 %
Symbian	0.5	0.2 %	6.5	4.2 %	-92.3 %
Others	N/A	0.0%	0.3	0.2 %	-100.0 %
Total	**236.4**	**100.0 %**	**156.2**	**100.0 %**	**51.3 %**

Vuoden 2012 toisen vuosineljänneksen ja vuoden 2013 toisen vuosineljänneksen välisenä aikana älypuhelinten toimitusmäärät kasvoivat IDC:n mukaan 51,3 %. Android-älypuhelinten markkinaosuus kasvoi 10,2 prosenttiyksikköä ja toimitusmäärä huimat 73,5 %. (IDC 2013.)

Applen osuuden pienentyminen selittyy osittain sillä, ettei iPhone 5 jälkeen ole tullut mitään merkittävää uutuutta markkinoille. Androidin osuutta kasvattaa Samsungin Galaxy S4 sekä muiden laitevalmistajien, kuten LG:n ja Huawein, Android-pohjaiset älypuhelimet. (IDC 2013.)

5.5 Android pelimarkkinoilla

Vaikka Android on ylivoimaisesti yleisin mobiilikäyttöjärjestelmä, tapahtuu mobiilikauppojen liikevaihdosta suurin osa, 76 %, Applen App Storessa. Suomalaisista mobiilipeleistä lähes kaikki ilmestyvät iOS:lle. Kummallakin alustalla voi kuitenkin saada aikaiseksi tuottavan liiketoiminnan, sillä USA:n markkinoilla iOS:lla on vahvat markkinat ja paremmat liikevaihdot käyttäjää kohden. Supercellin hittipelistä Clash of Clans:sta tehtiin Android-versio, joka on lokakuusta lähtien ollut saatavissa Google Play:stä. Tämä saattaa herättää muidenkin pelinvalmistajien mielenkiinnon Android-pelien kehittämiseen. (It-Viikko, 2013a.)

Yhtenä syynä pelinkehittäjien Android-vastaisuuteen saattaa olla tietoturvallisuus, sillä esimerkiksi F-Securen mukaan Android on turvattomin mobiilikäyttöjärjestelmä. F-Securen mukaan vuoden 2013 alkupuolella löytyi 358 Androidille tehtyä uutta haittaohjelmaperhettä. Todellisten haittaohjelmien määrä on siis moninkertainen, sillä haittaohjelmaperheen koodista syntyy useampia viruksia. Myös tietoturvayhtiö Symantec raportoi Androidille tehtyjen haittaohjelmien määrän rajusta kasvusta. Symantecin mukaan haittaohjelmia löytyy Google Play:stä, mutta suurin riski on kuitenkin kolmansien osapuolien verkkokaupoissa. Tunnettujen haittaohjelmien määrä on viimeisen vuoden aikana kasvanut 32 000:sta jopa 273 000 kappaleeseen. Samanaikaisesti haittaohjelmaperheiden määrä on kasvanut 69 prosentilla 121:sta 204 kappaleeseen. (It-Viikko, 2013b; It-Viikko 2013c.)

6 MOBIILIPELIEN KEHITYS

Mobiilipelien historia alkaa jo 70-luvulta, kun ensimmäisiä elektronisia kannettavia pelilaitteita alkoi ilmestyä markkinoille. Mattel Auto Race julkaistiin vuonna 1976 ja se sisälsi yhden pelin. 1980-luvulla Nintendo julkaisi suuren suosion saavuttaneet Game & Watch -pelilaitteet, joihin tehtiin paljon erilaisia pelejä kuten esimerkiksi Donkey Kong ja Super Mario Bros. Tässä opinnäytetyössä keskitytään enemmän nykypäivän mobiilipeleihin, joiden voidaan sanoa saaneen alkunsa, kun Nokia vuonna 1997 toi markkinoille 6110-mallissaan klassikoksi muodostuneen Snake-pelin. Vuonna 2005 Nokia arvioi pelin löytyvän noin 350 miljoonasta matkapuhelimesta. (Pelitieto 2009; Pelitutkimus 2009.)

1990-luvun lopussa WAP, Wireless Application Protocol, mahdollisti internetin käytön langattomasti matkapuhelimella, jolloin tuli mahdolliseksi ladata ja pelata pelejä verkossa. Kiinnostus palveluun oli vähäistä hitaiden tietoliikenneyhteyksien ja kankean pelattavuuden vuoksi. Vaikka kuluttajien kiinnostus oli vähäistä, uskottiin WAP- ja tekstiviestipeleillä olevan valtavasti potentiaalia, joten rahoittajat olivat valmiita sijoittamaan rahaa lupaaviin kehitysprojekteihin. Suomalainen pelikehitysstudio Riot Entertainment keräsi 21,5 miljoonan euron rahoituksen, mutta joutui vararikkoon kahden vuoden toiminnan jälkeen. Riot-E:n tuottama ja julkaisema yksinkertainen X-Men-peli saavutti kuitenkin suosiota. (Pelitutkimus 2009.)

WAP- ja tekstiviestipelien jälkeen matkapuhelimien kehittyessä pelejä alettiin tehdä Javalla ja BREW:lla, Binary Runtime Environment for Wireless, joista jälkimmäinen on Pohjois-Amerikassa suosioon noussut kehitysalusta. Pelejä kehitettiin myös suoraan Nokian tukemille Symbian S40- ja S60-sovellusalustoille. Java, BREW ja Symbian muodostuivat vakiintuneiksi teknologioiksi, joilla suurin osa mobiilipeleistä kehitettiin. Matkapuhelimien pelillisiin ominaisuuksiin panostettiin ensi kerran vuonna 2003, kun Nokia julkisti N-Gage-pelipuhelimen. Laajasta huomiosta huolimatta N-Gage ei kuitenkaan menestynyt odotetulla tavalla, sillä laitteessa oli useita käytettävyyteen liittyneitä suunnitteluvirheitä. N-Gage-pelit eivät myöskään ole saaneet suurta suosiota. Myöhemmin N-Gage-brändi on muutettu laitteesta rajapinnaksi Nokian N-sarjan älypuhelimille ja N-Gage-pelipalvelu N-Gage.com avattiin vuonna 2008. (Pelitutkimus 2009.)

6.1 Nykyisten pelialustojen ominaisuuksia

Matkapuhelimia on maailmassa paljon, useimmiten perheen jokaisella jäsenellä on oma puhelin käytössään. Matkapuhelin on tietokonetta helpompi hankkia esimerkiksi kehitysmaissa, sillä langattoman tietoliikenneverkon vaatima infrastruktuuri on helpompi rakentaa ja ylläpitää. Puhelimet eivät myöskään ole niin riippuvaisia paikoitellen epävakaasta sähkönsaannista, kuin tietokoneet. Puhelinten laaja levinneisyys tarjoaa kattavat markkinat mobiilipalveluille ja -peleille. (Pelitutkimus 2009.)

Puhelin on yleensä henkilökohtainen, ja koska se on kooltaan pieni ja kevyt, se kulkee koko ajan omistajan mukana, jolloin pelejä ja muita toimin-

toja voi käyttää halutessaan. Pelaamiseen ei tarvita erillisiä lisälaitteita tai johtoja, puhelin toimii sekä pelialustana että ohjauslaitteena. Puhelimen mukana tulee valmiina erilaisia sensoreita, joilla voidaan tallentaa tieto esimerkiksi käyttäjän sijainnista ja liikkumisesta. (Pelitutkimus 2009.)

Myytävänä on paljon erikokoisia laitteita ja laitevalmistajat pyrkivät hyödyntämään käytettävän tilan mahdollisimman tehokkaasti näytölle ja näppäimistölle, joka on usein nykyään virtuaalinäppäimistö. Näytön fyysisen koon kasvattamisen lisäksi kuvapintaa voidaan kasvattaa parantamalla näytön resoluutiota. Perinteinen näppäimistö saattaa aiheuttaa pelaajalle harmia, sillä pelitilanteessa näppäimistön näppäimet ovat usein liian pienet ja hankalasti käytettävissä. Koska puhelin toimii ensisijaisesti kommunikointivälineenä, sen fyysisiä näppäimiä ei poisteta käytöstä pelaamisen ajaksi, jolloin virhepainallusten mahdollisuus on suuri. Älypuhelimissa ja tableteissa pelit toimivatkin usein näyttöä koskettamalla tai laitetta kallistamalla. (Pelitutkimus 2009.)

Pelisuunnittelijan kannalta matkapuhelimissa on paljon mielenkiintoisia teknisiä ominaisuuksia, kuten erilaiset radiolähettimet kuten Bluetooth, WLAN, 3G ja 4G sekä kamera, mikrofoni ja erilaiset sisäänrakennetut sensorit, kuten GPS, kiihtyvyyssensori ja gyroskooppi. Radiolähettimien avulla voidaan olla yhteydessä toisiin pelaajiin ja pelipalvelimiin. Bluetooth-yhteydellä voidaan luoda väliaikainen yhteys lähistöllä oleviin pelaajiin. WLAN-, 3G- ja 4G-verkot tarjoavat nopeampia yhteyksiä internetiin, joskin niiden verkkokattavuudessa ja tehoissa on eroja. Kameran ja mikrofonin avulla pelaaja voi esimerkiksi tallentaa matkapuhelimella erilaisia asioita, jotka muunnetaan pelielementeiksi. Pelisuunnittelijat keräävät pelaajista ja näiden ympäristöstä tietoa erilaisten sensoreiden avulla. Tähän mennessä eniten käytetty sensoridata on ollut paikkatieto, jonka avulla pelaajien sijainti voidaan lähettää pelipalvelimille, jotka välittävät tietoa muille pelaajille. Gyroskoopin ja kiihtyvyyssensoreiden avulla matkapuhelimesta tehdään aktiivinen ohjauslaite, jonka asennot ja liikeradat voidaan jäljittää. Näiden tietojen avulla voidaan seurata, millaisia liikkeitä pelaaja tekee tai miten hän käyttää matkapuhelinta ohjauslaitteena. (Pelitutkimus 2009.)

6.2 Mobiilipelit nykypäivänä

Pelityyppien kirjo on yhtä laaja kuin tietokone- ja konsolipeleissä. Usein tietokoneelle tai pelikonsolille tuotetuista hittipeleistä tehdään yksinkertaisemmat versiot mobiilipelilaitteille. Selkeä ero varsinkin matkapuhelinpeleissä verrattuna tietokone- ja konsolipeleihin on se, että matkapuhelinpelejä pelataan usein ilman ääniä, koska pelaaja ei halua herättää ylimääräistä huomiota esimerkiksi linja-autossa tai junassa matkustaessaan. Toki mobiilipelilaitteilla pelataan myös kotona ja muissa paikoissa, joissa voidaan viettää ajallisesti pitkiäkin aikoja. Pelitieto-sivusto kertoo, että Nokian tekemän tutkimuksen mukaan matkapuhelimilla pelattiin keskimäärin n. 30 minuutin mittaisia pelisessioita. Tämä ei kuitenkaan tarkoita sitä, ettei pelejä pelattaisi liikkeellä ollessa sillä noin 60 % vastaajista kertoi pelaavansa mobiilipelejä matkapuhelimella ollessaan liikkeellä. (Pelitieto 2009.)

Nykyisin kannettavat pelilaitteet ja matkapuhelimet sisältävät kehittyneet verkko-ominaisuudet, joiden ansiosta laitteilla on usein nopea yhteys internetiin. Tämä mahdollistaa uudenlaisten pelien pelaamisen, sekä tietysti moninpelaamisen verkon yli. Moninpelaaminen ei kuitenkaan ole vielä niin yleistä mobiilipelilaitteilla kuin tietokoneilla ja pelikonsoleilla, mutta tilanne muuttuu koko ajan. Erityisesti Applen laitteille on julkaistu useita pelejä, joita voidaan pelata moninpelinä internetissä. Mikäli mobiilipelilaitteessa on internet-selain, voidaan sen avulla pelata useimpia selainpelejä joita on tarjolla sosiaalisen median verkkopalveluissa, esimerkiksi Facebookissa. (Pelitieto 2009.)

Jatkuva yhteys internetiin mahdollistaa asynkronisten pelien pelaamisen, jolloin peli pyörii taustalla riippumatta siitä onko pelaaja kirjautuneena peliin sisään vai ei. Asynkronisuus antaa pelaajalle mahdollisuuden osallistua peliin silloin kun parhaiten sopii, sillä pelaajan ei välttämättä tarvitse tehdä muuta kuin tarkistaa pelitilanne muutaman kerran päivässä ja tarvittaessa tehdä hieman muutoksia. Asynkronisessa moninpelissä osallistujien ei tarvitse pelata yhtä aikaa, vaan he voivat liittyä peliin ja poistua siitä omaan tahtiinsa ilman, että pelin kulku häiriintyy. Asynkronisessa moninpelissä pelaajat voivat olla vuorovaikutuksessa keskenään, vaikka kaikki osapuolet eivät olisikaan samanaikaisesti läsnä. (Pelitieto 2009; Pelitutkimus 2009.)

6.3 Hyvän mobiilipelin ominaisuuksia

Hyvän mobiilipelin tulee tukea käyttöliittymävaatimusten lisäksi myös verkottuneisuutta, sosiaalisuutta ja matkapuhelimelle ominaista käyttökulttuuria. Matkapuhelin on avoinna ollessaan yleensä aina verkossa ja se pystyy hyödyntämään erilaisia verkkoteknologioita. Mobiilipelejä suunniteltaessa törmätään erittäin hajautuneeseen laitekantaan, ja pelikehittäjät tekevät pelejä useille kymmenille, jopa sadoille laitealustoille, jotta riittävä ostovoima tavoitetaan. Pelikehittäjät joutuvat tekemään rajauksia tuettavien matkapuhelinmallien sekä verkko- ja sensoriteknologioiden välillä, sillä kaikkia malleja on lähes mahdotonta tukea. (Pelitutkimus 2009.)

Tietoliikenneverkkojen mahdollistamat moninpelit ovat olleet pitkään suosittuja konsoli- ja tietokonepeleissä, ja nykyisin älypuhelimilla ja tableteilla on mahdollista pelata mobiileja moninpelejä. Mobiilipeleissä samalla käyttäjätunnuksella voi usein pelata puhelimella, tabletilla ja internetissä. Moninpelejä pelatessa pelikumppanin löytyminen internetistä riippuu esimerkiksi siitä, kuka sattuu sillä hetkellä olemaan paikalla. Matkapuhelimeen voidaan tallentaa pelikumppaneiden yhteystietoja, jolloin verkosto kulkee pelaajan mukana. Nykyään mobiilipelit osaavat jo hyödyntää yhteystietolistoja pelaajien yhdistämisessä. Omasta puhelimesta tai vaikka Facebookista voi jakaa pelejä ja kutsuja pelitovereille lähettämällä viestin, jossa on mukana itse peli tai linkki peliin. Tätä tapaa kutsutaan itsestään leviäväksi jakelutieksi. (Pelitutkimus 2009.)

Puhelimen käyttökulttuuri erottaa mobiilipelit konsoli- ja tietokonepeleistä. Yleensä puhelin kulkee omistajansa mukana, sitä käytetään lyhyitä

hetkiä kerrallaan ja pelitilanteet voivat keskeytyä puhelimen muiden toimintojen, kuten puhelujen ja viestien vuoksi. Pelitilanteet voivat olla sellaisia, että käyttäjä ei välttämättä voi täysin keskittyä pelaamiseen, vaan hänen on samalla huomioitava ympäristö ja lähistöllä olevat ihmiset, jottei aiheuttaisi häiriötä pelaamisellaan. Tietyissä tilanteissa ja ympäristöissä matkapuhelimen käyttö ja siten myös pelaaminen on kielletty tai ainakin epätoivottavaa. (Pelitutkimus 2009.)

Menestyminen pelialalla ei ole itsestään selvyys, mutta muutamia asioita huomioimalla onnistumisen mahdollisuudet ovat paremmat. Epäonnistuminen täytyy tehdä nopeasti. Projektiin ei pidä kiintyä liikaa, vaan se pitää osata hylätä ajoissa ja ottaa opiksi virheistä. Yleisö ja siltä tullut palaute on erittäin tärkeää. Palautteen avulla peliä voidaan kehittää niin, että käyttäjät haluavat tietää mitä seuraavaksi tapahtuu ja ovat valmiita jatkamaan pelaamista. Pelillistämisellä, gamification, edistetään kuluttajien tai palvelunkäyttäjien osallistumista tiettyyn palveluun, esimerkiksi kanta-asiakaspisteiden keräämiseen, peleistä tuttujen ominaisuuksien avulla. Yhteistyökumppaneita kannattaa etsiä ulkomailta, suurin osa pelialan viennistä tapahtuu ulkomaille. Internetin ansiota markkinointikeinot ovat muuttuneet perinteisistä tavoista bloggaamisen, twiittaamisen ja muun sosiaalisen median pariin. Enää hyvään mainostamiseen ei välttämättä tarvita suuria rahasummia. (Kauppalehti 2013a; yxmanty 2012.)

6.4 Suomalaiset peliyhtiöt mobiilimarkkinoilla

Viimeisen parin vuoden aikana suomen pelialalla on tapahtunut huomattavaa kehitystä. Rovion ja Supercellin menestys on mullistanut alan liikevaihdon ja lisännyt työntekijöiden määrää. Tällä hetkellä työmarkkinoilla on huutava pula osaavista peliohjelmoijista, alalla arvioidaan olevan vapaana 600–700 työpaikkaa. Pelkästään Kotkan ja Kouvolan alueelle on syntynyt kymmeniä peliyrityksiä. Kymenlaakson maakunta käynnisti 1.6.2011 Kaakon peliklusteri -projektin, jonka tavoitteena on muun muassa kehittää alueen toimintaympäristöä pelialan harjoittamiseen sopivaksi. Projekti päättyy 31.12.2013. (Kymenlaakson liitto 2013; Talouselämä 2013, Tekniikka & talous 2013.)

Talouselämä on listannut Suomen eniten työllistävät peliyritykset. Seuraavaan kuvaan on laitettu listalta 10 eniten työllistävää yritystä.

Suomen suurimmat peliyritykset 2013

Yritys, perustamisvuosi	*työntekijöitä*
Rovio Entertainment, 2003	650
Supercell, 2010	106
Remedy, 1995	101
RedLynx, 2000	90
Sulake, 2000	70
Frozenbyte, 2001	52
Housemarque, 1995	44
Bugbear Entertainment, 2000	40
Digital Chocolate, 2000	n. 30
Kuuasema, 2004	29

Kuva 11 Talouselämän julkaisemia eniten työllistäviä peliyrityksiä

Syyskuussa 2012 Oululainen peliyhtiö Fingersoft julkaisi Androidille ilmaispelin Hill Climb Racing ja noin kuukautta myöhemmin pelistä julkaistiin myös iOS-versio. Peli on ollut alusta asti erittäin suosittu ja lokakuuhun 2013 mennessä peliä on ladattu 100 miljoonaa kertaa ja sillä on päivittäisiä pelaajia 46 miljoona ympäri maailman. (Kauppalehti 2013b.)

Pelifirma Crand Cru on aloitteleva peliyritys, jonka taustalla on kuitenkin vankkaa kokemusta pelikehityksestä. Perustajajäsenistä viisi tuli mobiiliyritys Mr. Goodlivingistä ja yksi jäsenistä tuli Sulakkeesta, joka tunnetaan Habbo-hotellin kehittäjänä. Grand Cru pyrkii hakemaan rahoittajikseen ulkomaisia sijoittajia ja saikin kesällä 2013 herätettyä median huomion saamalla 8,5 miljoonan euron uuden rahoituksen entisten lisäksi. Crand Cru kehittää pelejä Applen tableteille ja älypuhelimille. Tällä hetkellä kehitteillä on Supernauts-peli, jossa pelaaja rakentaa jäätiköiden sulamisen jälkeen avaruuteen asumiskelpoisia saarekkeita. Peli julkaistiin ensin testaustarkoituksessa Kanadassa ja nyt sen löytää jo App Storesta. (Ilta-Sanomat 2013; Helsingin Sanomat 2013.)

Supercell julkaisi vuonna 2012 Clash of Clans- sekä Hay Day -pelit. Molemmat pelit ovat olleet latauslistojen kärkisijoilla ja esimerkiksi Hay Day -peli nousi Yhdysvalloissa yhden viikonlopun aikana Top 10 -listalle. (MikroPC 2013.)

Rovio Mobilen kehittämä Angry Birds julkaistiin vuonna 2009. Sitä myytiin ensimmäisen puolen vuoden aikana yli 6,5 miljoonaa kappaletta. Myöhemmin peliä laajennettiin muille pelialustoille sopivaksi ja Rovion

oman verkkosivun mainoksen mukaan pelillä on jo 1,7 miljardia latausta. (Rovio 2013.)

6.5 Pelialan tilastoja Suomessa

2000-luvun nopeimmin kasvava viihdeteollisuuden haara on ollut peliteollisuus. Pelimyynnin arvioitiin vuonna 2012 olevan maailmanlaajuisesti noin 65 miljardia dollaria, jolloin se on ohittanut musiikin myynnin ja lähestyy elokuvien myyntiä. Viime vuosina peliteollisuuden vuosittainen liikekasvu on saattanut olla jopa yli 50 prosenttia.

Kuva 12 Peliteollisuuden liikevaihdon kasvu Suomessa

Suomessa peliteollisuudesta on tullut merkittävä vientiartikkeli. Pelimarkkinoiden kotimaan osuus on pieni ja siksi yli 90 prosenttia suomalaisesta pelialan tuotannosta päätyy vientiin. Kokonaisliikevaihdossa huomioidaan myös sijoitus- ja liiketoiminta, joka tapahtuu pelinkehityksen ja pelipalveluiden ulkopuolella, esimerkiksi yrityskaupat, oheistuote- ja lisenssimyynti sekä toimialalle tehdyt investoinnit. Vuonna 2012 Neogamesin mukaan pelinkehityksen ja pelipalveluiden liikevaihto oli 250 miljoonaa euroa, mutta kokonaisliikevaihto oli arvioilta noin 350 miljoonaa euroa. (Neogames, 2013, Yle, 2013.)

Alun perin Neogames arvioi Suomen pelinkehityksen ja pelipalveluiden liikevaihdon kasvavan yli 200 % vuonna 2013, noin 800 miljoonaan euroon. Tilanne kuitenkin muuttui huomattavasti, kun Japanilaiset SoftBank ja GungHo ostivat Supercellistä 51 % huimalla 1,1 miljardin euron kauppahinnalla. Tämän mukaan vuoden 2013 arvio liikevaihdon kokonaisarvosta nousee noin kahteen miljardiin euroon. Neogamesin johtaja KooPee Hiltunen arvioi, ettei pelialan kasvun päätepiste ole vielä lähimaillakaan. (Neogames, 2013; Yle, 2013.)

6.6 Pelialan tilastoja Yhdysvalloissa

Entertainment Software Association -sivuston mukaan 36 % pelaajista käyttää pelaamiseen älypuhelinta ja 25 % käyttää pelaamiseen kannettavaa

tai tablet-tietokonetta. Vuodesta 2005 vuoteen 2009 mennessä viihdeohjelmistoalan vuotuinen kasvoi yli 10 prosenttia, samalla kun Yhdysvaltojen talous kasvoi alle kaksi prosenttia. Vuonna 2009 viihdeohjelmistoala kasvatti Yhdysvaltain bruttokansantuotetta 4,9 miljardilla dollarilla. (esa, 2013.)

7 PELI

Keväällä 2013, ennen tämän opinnäytetyön aloittamista, käynnistyi mobiilipeliprojekti, jossa on mukana useita henkilöitä suunnittelemassa ja luomassa peliä. Pelimaailma koostuu tehtävistä ja välietapeista. Yhden pelialueen teemat ovat samanlaisia, esim. yhteenlaskuja, seuraavassa miinuslaskuja, sitten kertolaskuja jne. Pelimaailma esitetään karttana. Yhdellä pelialueella on 5-8 asteittain vaikeutuvaa tehtävää, joiden jälkeen tulee jokin peli.

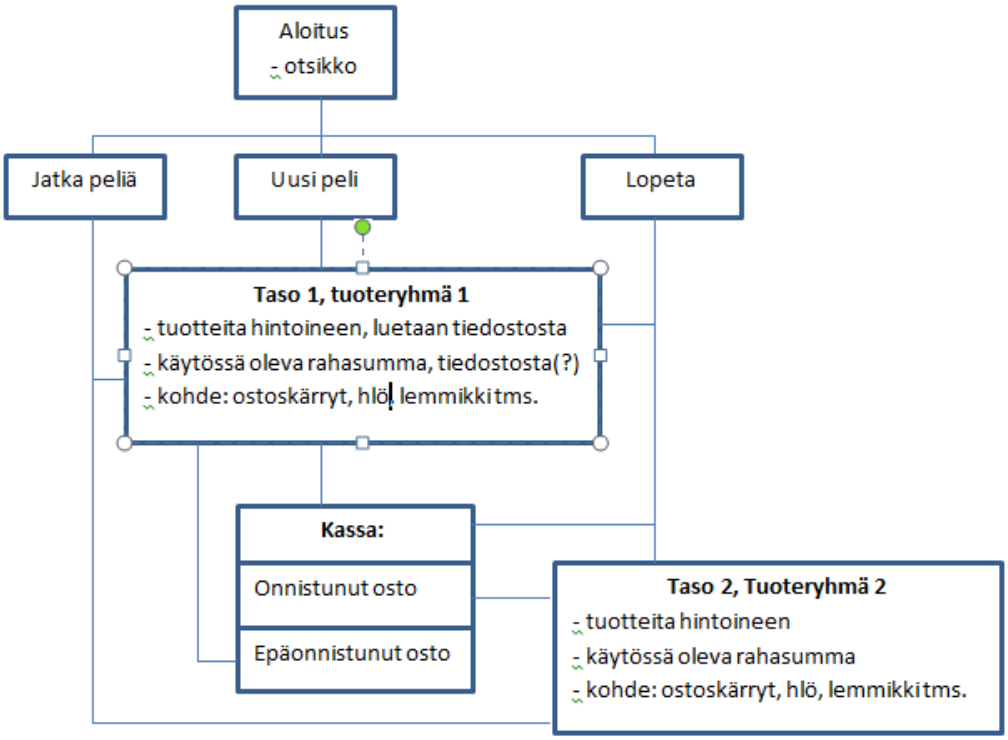
7.1 Tavoite

Toimeksiantajan tavoitteena on saada tuotantoon toimiva, kiinnostava ja opettava esikouluikäisille suunnattu mobiilipeli, jolla tavoitellaan myös taloudellista hyötyä ja jatkuvuutta. Opinnäytetyön tavoitteena on suunnitella ja toteuttaa toimiva, 2–3 tasoinen, esimerkkigrafiikoilla toimiva tehtävä mobiilipeliin käyttämällä Corona SDK -sovelluskehitysalustaa.

7.2 Idea ja rakenne

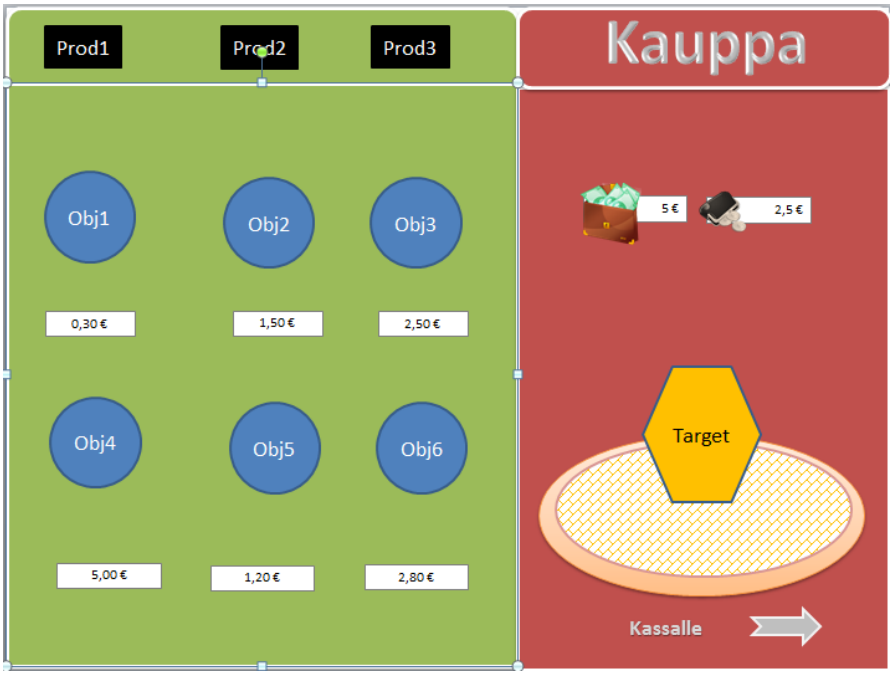
Pelin ideana on ostaminen. Ostaa voidaan esimerkiksi elintarvikkeita, henkilölle vaatteita tai tavaroita, lemmikille leluja tai asusteita. Pelissä on kolme tuoteryhmää, joista ensin vain yksi tuoteryhmä on aktiivinen. Pelaaja valitsee tuotteen tai useita tuotteita rahatilanteensa mukaan. Tässä vaiheessa ei puututa siihen millä perusteella pelaaja saa lisää rahaa käyttöönsä. Rahaa voi tulla esimerkiksi aikaisempien pelien tulosten mukaan tai peli voi arpoa pelaajalle rahasumman. Tuote raahataan esimerkiksi ostoskärryyn tai henkilön tai lemmikin päälle. Kun ostokset on tehty, siirrytään kassalle.

Kassalla pelaaja näkee lopullisen summan, jolloin pelaajan täytyy valita lompakosta sopiva rahasumma, joko tasaraha tai enemmän. Peli palauttaa yli menevän rahasumman käyttäjän lompakkoon. Onnistuneella ostotapahtumalla pääsee siirtymään seuraavalle tasolle. Mikäli budjetti ylittyi, pelaaja voi yrittää uudestaan. Seuraavalla tasolla aktiivisena on uusi tuoteryhmä, josta löytyy nyt parempia ja kalliimpia tuotteita.

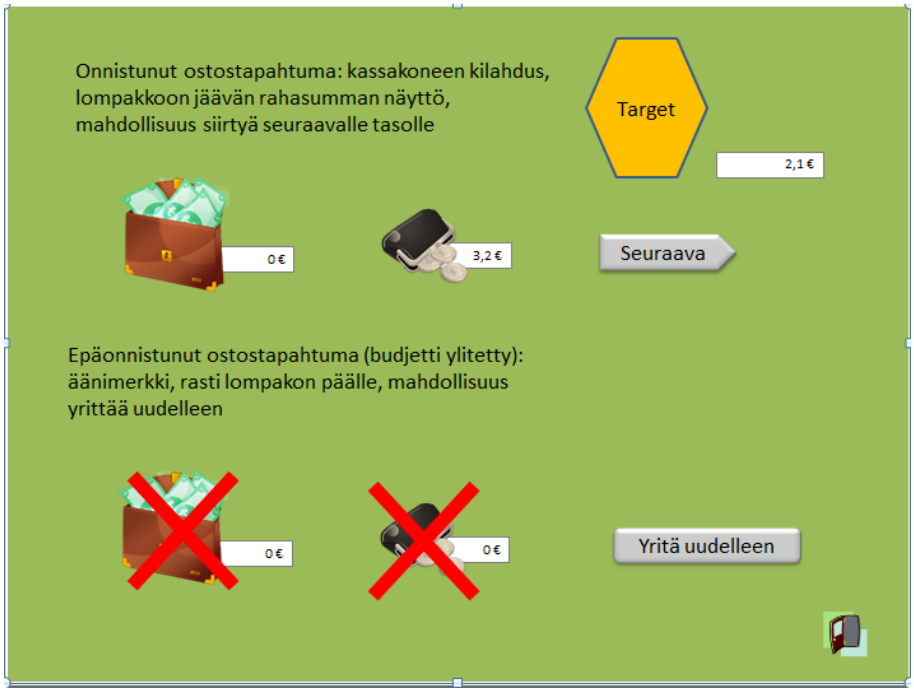


Kuvio 1 Pelin rakenne

Kuva 13 menuScenen suunnitelma



Kuva 14 gameScenen suunnitelma



Kuva 15 checkoutScene suunnitelma

Kuva 16 endScenen suunnitelma

7.3 Toteutus

Ennen ohjelmoimisen aloittamista peli tulee suunnitella huolellisesti ja tehdä määrittelydokumentti, jossa kuvataan mahdollisimman tarkasti pelin tavoitteet, rakenne, tarvittavat hahmot ja muu grafiikka, käytettävyys sekä äänimaailma. Opinnäytetyöstä ei laadittu määrittelydokumenttia, sillä tavoitteena on saada aikaiseksi esimerkkigrafiikoilla toimiva pelin prototyyppi. Pelin yleiset linjaukset sovittiin toimeksiantajan kanssa sähköpostitse ja hän antoi opinnäytetyön tekijälle melko väljät raamit toteutuksen suhteen.

Kun Coronassa aloitetaan uusi projekti, Corona tekee valmiiksi build.settings-, config.lua- ja main.lua-tiedostot. Näiden tiedostojen avulla pelin rakentaminen aloitetaan. Build.settings-tiedostossa määritellään käytetäänkö vaaka- vai pystysuuntaista alustaa vai tuetaanko molempia, Config.lua-tiedostossa määritellään esimerkiksi pelin oletuskoko sekä mahdollinen skaalautuvuus ja main.lua-tiedostossa aloitetaan varsinainen ohjelmointi.

Opinnäytetyössä ennen ohjelmoinnin aloittamista suunniteltiin myös mistä osista, sceneistä, peli muodostuu. Scene on yleensä pelissä näkyvä, yksilöllinen näyttö, eli screen. Scenejä hallinnoidaan Coronan tarjoamalla Storyboard API:lla. Rajapinta tarjoaa scenejä varten määritellyn koodirakenteen, määrittelee scenen tarvitsemia elinkaaren tapahtumia ja selventää scenejen hallintaan tarvittavia toimintoja. Tähän peliin suunniteltiin menuScene, gameScene, checkoutScene ja endScene.

```
local storyboard = require( "storyboard" )
local scene = storyboard.newScene()

-- Called when the scene's view does not exist.
function scene:createScene(inEvent)
end

-- Called BEFORE scene has moved on screen.
function scene:willEnterScene(inEvent)
end

-- Called AFTER scene has moved on screen.
function scene:enterScene(inEvent)
end

-- Called BEFORE scene moves off screen.
function scene:exitScene(inEvent)
end

-- Called AFTER scene has moved off screen.
function scene:didExitScene(inEvent)
end

-- Called prior to the removal of scene's "view" (display group).
function scene:destroyScene(inEvent)
end

-- Add scene lifecycle event handlers.
scene:addEventListener("createScene", scene);
scene:addEventListener("willEnterScene", scene);
scene:addEventListener("enterScene", scene);
scene:addEventListener("exitScene", scene);
scene:addEventListener("didExitScene", scene);
scene:addEventListener("destroyScene", scene);

return scene;
```

Kuva 17 Storyboard Scenen perusrakennne. (Zammetti 2013.)

Jokainen scene tallennetaan omana tiedostonaan ja niille annetaan sceneä vastaava nimi. Esimerkiksi menuScene.lua, gameScene.lua ja niin edelleen. Storyboardin aktivoinnin jälkeen Corona käy läpi scenen tapahtumakuulijat. Tässä esimerkissä näytettiin kaikki tapahtumat, joihin scene voi vastata, mutta käyttäjä itse päättää mitkä tapahtumista ovat tarpeellisia omassa tapauksessa. Tätä samaa rakennetta kopioidaan ja liitetään osiosta toiseen ja tehdään rakenteeseen vain osiokohtaisia muutoksia. (Zammetti 2013)

Main.lua on tiedosto, josta jokaisen sovelluksen suorittaminen alkaa. Pienissä peleissä kaikki koodi voidaan tehdä suoraan main.lua-tiedostoon. Opinnäytetyössä peli on jaettu eri sceneihin ja siksi täytyykin miettiä mitä kaikkea main.lua-tiedostoon sijoitetaan. Perussääntönä voisi pitää sitä, että yleiset, globaalit toiminnot, sijoitetaan tähän tiedostoon. Pelin nopeuden kannalta on parempi, mitä vähemmän koodissa on yleisiä funktioita tai yleisiä muuttujia, ohjelmoijien tulisi mahdollisimman paljon käyttää paikallisia, lokaaleja funktioita ja muuttujia. Opinnäytetyössä main.lua-tiedostoon on muun muassa sijoitettu yleiset tuonnit json- ja storyboard-kirjastoille, tieto käytettävästä tuoteryhmästä, pelin tallentaminen ja lataaminen sekä siirtyminen seuraavaan sceneen.

Nimensä mukaisesti menuScene.lua-tiedosto sisältää aloitusmenun. Menussa on kaupan logo sekä tekstit uusi peli, jatka peliä ja lopeta. Tekstiä napauttamalla aktivoidaan kyseinen toiminto. MenuScene.lua-tiedostoon sijoitetaan storyboardin perusrakenne, jota muokataan tarpeisiin sopivaksi. Tässä tapauksessa funktioon createScene sijoitetaan kaikki näytöllä näkyvät objektit ja määritellään teksteille tapahtumankäsittelijät. Funktiossa destroyScene määritellään vielä, mitkä objektit näytöltä tuhotaan, kun scene poistettaan. Jokaisen scenen kohdalle koodiin tehdään vielä print-komento, jonka avulla terminaaliin tulostetaan tekstiä helpottamaan mahdollisten virheiden paikannusta. Print-komennon avulla voidaan selvittää mitä tapahtuu ja mihin asti koodi suoriutuu.

GameScene.lua-tiedosto sisältää varsinaisen pelin. Tätä osiota varten luotiin kuvankäsittelyohjelmalla muutamia yksinkertaisia objekteja: painikkeet tuoteryhmiä varten, erimuotoisia objekteja tuotteita varten, kohteen johon ostetut tuotteet sijoitetaan ja lattiaa kuvaavan ellipsin, ettei kohde leijuisi ilmassa. Osio on jaettu vasempaan ja oikeaan puoliskoon. Vasen puolisko näyttää tuoteryhmät, joita on yhteensä kolme ja vain tuoteryhmä1 on alussa aktiivinen. Vasemmalla puolella näkyvät myös ostettavat tuotteet ja niiden hinnat. Alun perin tuotetiedot oli tallennettu products.json-tiedostoon, josta ne oli tarkoitus lukea gameScenessä olevaan tauluun. Tätä koodinpätkää ei kuitenkaan saatu toimimaan ja lopulta tuotteet ja niiden hinnat sijoiteltiin sceneen samalla tavalla, kuin menuScenen objektit. Oikealla puolella näkyy lompakossa oleva rahamäärä, kohde ja lattia sekä painike, josta siirrytään kassalle.

Vasemmalta puolelta tuote on tarkoitus raahata hiirellä kohteen päälle. Alun perin tarkoituksena oli, että tuotteen voisi ostaa halutessaan useamman kerran, mutta nyt tuote poistuu valikoimasta. Tuotteen oli tarkoitus näkyä kohteessa, esimerkiksi ostoskärryissä tai lemmikin päällä, mutta tuote siirtyi kohteen alle. Koska en keksinyt keinoa, jolla näytöllä olevien objektien järjestystä voi vaihtaa, siirsin kohteen sijaintia koodissa, jonka jälkeen tuote jäi kohteen päälle. Raahaaminen ei toimi siten, kuin tarkoittanut. Tuotetta raahatessa, se vain yhtäkkiä on kohteen päällä. Raahaaminen ei toimi siten, kuin tarkoittanut. Tuotetta raahatessa, se vain yhtäkkiä on kohteen päällä. Kun tuotteen raahaa kohteen päälle, ohjelma laskee lompakon rahatilanteen, mutta ei näytä sitä pelaajalle. Alun perin käyttäjän oli tarkoitus pystyä ostamaan useampia tuotteita, kunhan vain raha riittää, mutta nyt rahatilanne lompakossa lasketaan oikein vain silloin, kun käyttäjä ostaa yhden tuotteen.

Kun pelaaja omasta mielestään on ostanut ne tuotteet, joihin hänellä on tarpeeksi rahaa, hän siirtyy kassalle, jolloin pelissä siirrytään checkoutSceneen. CheckoutScenessä näkyy lompakon nykyinen tilanne. Mikäli pelaaja ei ylittänyt saldoaan tai mikäli lompakkoon jäi vielä rahaa, hän voi halutessaan jatkaa seuraavalle tasolle tai päättää pelin. Mikäli pelaaja ylitti saldonsa, hän voi yrittää uudelleen tai päättää pelin.

7.4 Testaus

Koodin vajavaisuuden ja ajanpuutteen vuoksi, ohjelmaa ei testattu.

7.5 Pelin kehittäminen ja jatkuvuuden takaaminen

Monissa peleissä on jo mukana osto-ominaisuus, jolla on tarkoitus edistää pelin kulkua esimerkiksi ostamalla kehittyneempiä varusteita strategiapeleissä ja kalliimpia eli enemmän hyötyä tuottavia hyödykkeitä vaikkapa Sims-pelissä. Pelkkänä ostamiseen keskittyvänä pelinä idea on hyvä ja kehityskelpoinen, kunhan vain keksitään kohderyhmälle sopivat ostettavat tuotteet, keksitään hyvä juoni, mihin ostotapahtuma perustuu ja pohditaan tavoite, johon ostamisella pyritään. Jos juoni ja tavoite saadaan mahdollisimman houkuttavaksi, voidaan peliin tarvittaessa kehittää jatko-osia, jolla pelaajien mielenkiinto säilyy tai jopa kasvaa. Hyvänä esimerkkinä toimii Sims-peli, johon tulee jatkuvasti uusia lisäosia. Lisäosa itsessään tuo peliin vain hieman lisää uusia mahdollisuuksia, esimerkiksi vuoden aikojen vaihtelut tai mahdollisuuden hankkia lemmikkejä. Lisäosan mukana tulee kuitenkin paljon uutta tavaraa ostettavaksi ja tämä on yksi houkutin lisäosan ostamiseksi.

Mikäli kohderyhmänä säilytetään esikouluikäiset lapset, joudutaan pohtimaan miten sosiaalisuus ja verkottuneisuus voitaisiin ottaa peliin mukaan. Lapset todennäköisesti haluaisivat jakaa pelikokemuksiaan ja ostotapahtumiaan verkossa nuorten ja aikuisten tapaan, mutta läheskään kaikilla esikouluikäisillä ei ole käytössään Facebookia tai mitään muuta yhteisöpalvelua. Esikouluikäiset kohderyhmänä aiheuttaa sen, että pelin käyttöliittymää ja käytettävyyttä joudutaan pohtimaan normaalia enemmän. Tekstejä pitää olla mahdollisimman vähän ja käytettävissä ikoneissa ja kuvissa ei saisi olla tulkinnanvaraa. Esikouluikäisten ajatusmaailma on ihan erilainen nuoriin ja aikuisiin verrattuna, joten ennen ostopelin suunnittelua täytyisi tehdä selvitystyötä siitä, millaisista asioista lapset ovat kiinnostuneita. Tulokset saattavat olla yllättäviä.

Opinnäytetyönä tehty koodi ei sovellu jatkokehitykseen. Ohjelmoijan taidot eivät olleet lainkaan riittävät sovelluksen kehittämistä ajatellen. Opinnäytetyön koodi on alkeellista ja yksinkertaista, se toimii vain osittain eivätkä tehdyt ratkaisut ole jatkokehityksen kannalta järkeviä.

8 OMAN OPPIMISEN ARVIOINTI

Valitsin opinnäytetyöni sellaisesta aiheesta, joka oli itselleni entuudestaan tuntematon tai ainakin hyvin alkeellisella tasolla. Ajatuksena oli opinnäytetyötä tehdessä oppia ohjelmoinnin perusteita. Koulutuksen ajalta mielenkiintoisimmat kurssit ovat liittyneet ohjelmointiin tai webbisivujen suunnitteluun ja arvelin tämän mielenkiinnon siivittävän opinnäytetyön valmistumista.

Opinnäytetyötä tehdessä perehdyin melko tarkkaan Corona SDKn, Luan ja Androidin historiaan ja ominaisuuksiin. Teoriaosuuden rajaaminen tuotti

hieman vaikeuksia, sillä jouduin miettimään esimerkiksi miten erottelisin Corona SDK:n asiat ja Luan. Molemmissa puhutaan samasta ohjelmointikielestä, mutta Corona SDK toimii tässä tapauksessa työvälineenä, joka mahdollistaa Luan hyödyntämisen eri alustoille suunnitelluissa sovelluksissa. Lua.org sivustolla tietoja oli hajautettu moneen paikkaan ja kerrottu ranskalaisilla viivoilla satunnaisessa järjestyksessä. Tästä sekamelskasta oli vaikea muodostaa yksinkertaista selostusta Luan ominaisuuksista. Yllättäen englanninkielisen tekstin kääntäminen muodostui kuviteltua vaikeammaksi tehtäväksi. Käännöksissä suora suomennos ei useinkaan avannut sisältöä millään tavalla, päinvastoin, suomennokset olivat usein aivan järjettömiä, vailla minkäänlaista kunnon tarkoitusta. Ennen kääntämistä olisi pitänyt osata ”puhua ohjelmointia” eli olisi pitänyt tietää alan suomenkielinen termistö. Termistössä tuli myös jonkin verran uusia suomenkielisiä sanoja, joiden tarkoituksen jouduin selvittämään. Pahin pelkoni onkin, että tekstissä on asiavirheitä, jotka johtuvat kääntämisen ja sanaston vaikeudesta. GitHubiin olin tutustunut aiemmin ohjelmoinnin projektityön yhteydessä, mutta silloin GitHubia käytettiin komentoikkunan kautta. Graafinen, helposti käytettävä käyttöliittymä oli positiivinen yllätys. Tietovarastonkin sai luotua vain vetämällä opinnäytetyö-kansion ikkunaan eikä tarvinnut muistella komentoja.

Opinnäytetyön ehdottomasti vaikein asia oli ohjelmoiminen. Ohjelmoitava peli itsessään ei ollut monimutkainen eikä varsinaisesti varmaankaan vaatinut valtaisia ohjelmointitaitoja. Oma ohjelmointikokemukseni perustuu vain ja ainoastaan koulun aikana käytyihin ohjelmointi-kursseihin, joten minulla ei ollut tarpeeksi rutiinia ja kokemusta tähän tehtävään. Koodia sain aikaiseksi vain lukemalla ohjelmointioppaita ja katselemalla erilaisia opetusvideoita ja lukemalla muiden ohjelmoijien laatimia tutoriaaleja. Tavoite, eli toimiva pelikoodi, jäi saavuttamatta ja lopputuloksena on koodia joka osittain toimii, mutta ratkaisut eivät ole jatkokehityksen kannalta järkeviä. Harmillisinta oli ehkä se, että kun omasta mielestäni olin saanut aikaiseksi toimivaa koodia, huomasin olevani väärässä, enkä millään tavalla saanut selville koodissa olevaa virhettä. Terminaaliin tulostuvat virheilmoitukset aiheuttivat vain ärsytystä, kun niissä viitattiin esimerkiksi virheeseen rivillä, jossa ei ollut koodia ollenkaan ja niin edelleen. Ihmiset oppivat asioita eri tavalla ja minun pitäisi ensin osata perusteet hyvin, ennen kuin voin soveltaa niihin uutta tietoa. En voi verrata Corona SDK:ta muihin sovelluskehitysalustoihin, mutta mielestäni se on tarkoituksenmukainen ja pidän sitä myös hyvänä sovelluksena aloittelevalle ohjelmoijalle. Aloittelijan kannattaa aloittaa perusteista ja pienistä ohjelmista ja sen jälkeen varmasti pääsee hyvin kehittymään.

Aikataulutus meni pieleen opinnäytetyötä tehdessä. Raporttiin tulevan tekstin tuottaminen sujui normaalia hitaammin aiemmin mainitsemieni käännösvaikeuksien takia. Osasin etukäteen varautua siihen, että ohjelmoiminen tulee viemään paljon aikaa, mutta hidas eteneminen on silti ollut yllättävän hidasta. Loppua kohti jouduin entistä enemmän tyytymään kompromissiratkaisuihin, jotta saisin palautettua edes kokonaisen pelirungon, vaikka se ei kaikilta osin toimikaan. En ota valokuvia ja siksi en myöskään käytä kuvankäsittelyohjelmia. Olen huomannut, että töiden tekeminen on hankalaa ilman peruskuvakäsittelytaitoja ja opinnäytetyössä

se kävi ilmi tehdessäni pelin grafiikkaa. Onneksi en joutunut tekemään oikeita tuotteita, pelissä käytettävien perusobjektienkin tekeminen oli aluksi hidasta. Suunnitelmassa olisi kannattanut tehdä jonkinlainen aikajana toteutettavia työvaiheita varten ja jokaista työvaihetta kohden joustoa olisi annettu vain muutama päivä. Jos tuota aikajanaa olisi noudattanut, opinnäytetyö olisi valmistunut paremmin aikataulun mukaan.

Vaikka en opinnäytetyön oheistuotteena oppinutkaan ohjelmoimaan, sain kuitenkin itselleni arvokasta tietoa omista rajoistani ja kiinnostuksen kohteista. Koulun aloittaessani olin erittäin kiinnostunut ohjelmoimisesta ja pidin sitä yhtenä mahdollisena tulevana ammattinani, mutta nyt arvelen, ettei näin tule käymään. Matkaa on liian paljon kuljettavana, jotta saisi perusteet ja todellisen ammattitaidon haltuun. Näin suuri ammatinvaihdos vaatisi aktiivista paneutumista ohjelmoinnin harjoitteluun ja siitä pitäisi tulla yksi vapaa-ajan harrastus. Opinnäytetyön ansioksi voidaan kuitenkin lukea se, että havahduin viimeinkin kuvankäsittelyn tarpeellisuuteen ja olen jo ilmoittautunut Kuvankäsittelyn alkeet -kurssille päivittääkseni tietoni.

9 YHTEENVETO

Opinnäytetyössä tärkeimmät kysymykset olivat mikä on Corona SDK, miten suunnitellaan ja toteutetaan opetuksellinen esikouluikäisille suunnattu mobiilipeli Corona SDK:lla ja millaisilla asioilla saadaan toteutuneelle pelille jatkuvuutta?

Opinnäytetyötä tehdessä tutustuttiin Corona SDK:hon ja Lua-ohjelmointikieleen. Nämä molemmat sopivat hyvin työkaluiksi mobiilipelin toteuttamiseen. Lua on helppo ohjelmointikieli yksinkertaisen rakenteensa vuoksi ja siitä on saatavilla monia oppaita ja tutoriaaleja, joten aloittelijakin pääsee tuottamaan toimivaa koodia. Corona SDK toimii työvälineenä, joka mahdollistaa Luan hyödyntämisen eri alustoille suunnitelluissa sovelluksissa.

Varsinainen pelin suunnittelu tapahtuu ihan muilla tavoilla, esimerkiksi vanhanaikaisesti kynää ja paperia käyttämällä, tekstinkäsittelyohjelmalla, kaavioilla tai muilla suunnittelijan tarvitsemilla ohjelmilla. Suunnittelussa pitäisi tehdä määrittelydokumentti, jossa kuvataan mahdollisimman tarkasti pelin tavoitteet, rakenne, tarvittavat hahmot ja muu grafiikka, käytettävyys sekä äänimaailma. Suunnittelussa täytyy huomioida mobiililaitteiden laaja valikoima sekä niiden vaihtelevat koot ja ominaisuudet ja tehdä tarvittavia rajauksia mallien ja teknologioiden välillä. Pelin suunnittelu on vaativa prosessi, josta saisi tehtyä oman opinnäytetyönsä

Mobiilipelin menestykseen ja jatkuvuuden takaamiseen vaikuttavat monet eri asiat. Verkottuneisuus hyödyntää tietoliikenneverkon ominaisuutta jakaa ja vastaanottaa informaatiota. Sosiaalisuus näkyy moninpelien suosiossa, pelitulosten jakamisessa sosiaalisessa mediassa sekä mainonnan lisääntymistä sosiaalisen median kanavien kautta. Yleisöltä tullut palaute on tärkeä osa pelin jatkokehityksessä.

LÄHTEET

AfterDawn, 2013. Puhelinvertailu. Viitattu 18.11.2013.
http://www.puhelinvertailu.com/uutiset.cfm/2013/08/09/androidin_kaytto_kaantyi_laskuun_usa_ssa_-_menestys_jatkuu_muualla

Android, 2013a. Viitattu 25.9.2013.
http://www.android.com/kitkat/index.html

Android, 2013b.Viitattu 25.9.2013.
http://www.android.com/about/

Android, 2013c. Viitattu 25.9.2013.
http://source.android.com/source/index.html

Android, 2013d. Viitattu 17.11.2013.
http://developer.android.com/about/dashboards/index.html

Android, 2013e. Viitattu 17.11.2013.
http://developer.android.com/guide/practices/screens_support.html

Android, 2013f. Viitattu 25.9.2013.
http://developer.android.com/about/index.html

Business Insider, 2013. Tech. How Android Grew To Be More Popular Than The iPhone. Viitattu 25.9.2013.
http://www.businessinsider.com/history-of-android-2013-8?op=1

Coronalabs, 2013. Products. Viitattu 18.6.2013.
http://www.coronalabs.com/products/corona-sdk/

Corona Docs, 2013a. Viitattu 13.11.2013.
http://docs.coronalabs.com/api/library/display/newImage.html

Corona Docs, 2013b. Viitattu 13.11.2013.
http://docs.coronalabs.com/guide/basics/configSettings/index.html

Entertainment Software Association. 2013. Facts. Viitattu 2013.
http://www.theesa.com/facts/econdata.asp
http://www.theesa.com/facts/pdfs/ESA_EF_2013.pdf

Esitykset. Viitattu 12.11.2013.
http://linux.ictlab.kyamk.fi/esitykset/2006_2/versionhallinta.odp

Git. 2013a. Viitattu 12.11.2013.
http://git-scm.com/

Git. 2013b. Small and fast. Viitattu 12.11.2013
http://git-scm.com/about/small-and-fast

Google Play, 2013. Viitattu 13.11.2013.

https://support.google.com/googleplay/android-developer/answer/113468?hl=fi

IDC Analyze the Future. Viitattu 25.9.2013.
http://www.idc.com/getdoc.jsp?containerId=prUS24257413

Helsingin Sanomat. 2013. Suomalainen pelifirma Grand Cru nappasi suurrahoituksen. Viitattu 26.11.2013.
http://www.hs.fi/talous/a1375068286409

Ilta-Sanomat. 2013. Miljoonafloppi oli mobiilipelien Suomi-komeetalle kova koulu. Viitattu 26.11.2013.
http://www.iltasanomat.fi/digi/art-1288620436699.html

It-Viiko. 2013a. Siirtävätkö suomalaiset mobiilipelaamisen valtikan iOS:ltä Androidille? Viitattu 25.11.2013.
http://www.itviikko.fi/uutiset/2013/09/27/siirtavatko-suomalaiset-mobiilipelaamisen-valtikan-ioslta-androidille/201313433/7

It-Viikko. 2013b. Google Play on liian vaarallinen tavalliselle käyttäjälle. Viitattu 25.11.2013.
http://www.itviikko.fi/tietoturva/2013/09/26/google-play-liian-vaarallinen-tavalliselle-kayttajalle/201313373/7

It-Viikko. 2013c. Varo pieniä Android-kauppoja. Viitattu 25.11.2013.
http://www.itviikko.fi/tietoturva/2013/10/30/tietoturvayhtio-varo-pienia-android-kauppoja/201315115/7

Kauppalehti. 2013a. Kopioi nämä opit pelifirmoilta ja menestyt. Viitattu 3.12.2013.
http://www.kauppalehti.fi/omayritys/kopioi+nama+opit+pelifirmoilta+ja+menestyt/201305430159

Kauppalehti. 2013b. Oululainen peliyhtiö rikkoi sadan miljoonan rajapyykin. Viitattu 25.11.2013.
http://www.kauppalehti.fi/omayritys/oululainen+peliyhtio+rikkoi+sadan+miljoonan+rajapyykin/201310535349

Kymenlaakson liitto. 2013. Kaakon peliklusteri. Viitattu 26.11.2013.
http://www.kymenlaakso.fi/suunnittelu_ja_kehittaminen/Rahoitus/issue_show.jsp?issueId=4815

Linux-Aktivaattori. ViikonValo. GitHub. Viitattu 12.11.2013.
http://l-a.fi/GitHub

Lua, 2013a. About. Viitattu 18.6.2013.
http://www.lua.org/about.html

Lua, 2013b. Viitattu 17.11.2013.
http://www.lua.org/license.html

Lua, 2013c. Viitattu 17.11.2013.
https://sites.google.com/site/marbux/home/where-lua-is-used

Lua, 2013d. Viitattu 17.11.2013.
http://lua-users.org/lists/lua-l/2007-11/msg00248.html

Luong, S. 2013. Reseptinhallintaohjelma
Metropolia Ammattikorkeakoulu. Automaatiotekniikka. Insinöörityö

Mac Developer Program. Viitattu 13.11.2013.
https://developer.apple.com/programs/mac/

Mobiili, 2013. Viitattu 18.6.2013.
http://mobiili.fi/2013/05/09/microsoft-ostamassa-android-laitteista-luopuvan-nook-median-miljardikaupalla/

Neogames. 2013. Tietoa toimialasta. Viitattu 26.11.2013.
http://www.neogames.fi/tietoa-toimialasta/

Pelitieto. Pelien peruskurssi, 2009. Viitattu 25.11.2013.
http://pelitieto.net/case-mobiilipelaaminen/

Pelitutkimuksen vuosikirja. 2009. Toim. Jaakko Suominen et al. Tampereen yliopisto. Sivut 67–81. Viitattu 25.11.2013.
http://www.pelitutkimus.fi/wp-content/uploads/2009/08/ptvk2009-06.pdf

Reaktor, 2013. Viitattu 12.11.2013.
http://reaktor.fi/osaaminen/hajautettu_versionhallinta/

Rovio, 2013. Viitattu 1.12.2013.
http://www.rovio.com/en/advertise

Vapise, Farmville – suomalainen Hay Day nousi hetkessä iPadin top10-listalle. Viitattu 1.12.2013.
http://www.mpc.fi/kaikki_uutiset/vapise+farmville++suomalainen+hay+day+nousi+hetkessa+ipadin+top10listalle/a818849

Talouselämä. 2013. Pienikin voi menestyä pelialalla - Frogmind on kahden miehen kultasuoni. Viitattu 26.11.2013.
http://www.talouselama.fi/uutiset/pienikin+voi+menestya+pelialalla++frogmind+on+kahden+miehen+kultasuoni/a2190319

Tekniikka & talous. 2013. Kuoleeko Suomen peliala kehtoonsa? Viitattu 26.11.2013.
http://www.tekniikkatalous.fi/talous/kuoleeko+suomen+peliala+kehtoon-sa+ss+koulutus+ei+pysty+vastaamaan+alan+tyovoimapulaan/a940056

Visual, 2013. A brief history of android. Viitattu 25.9.2013.
http://visual.ly/brief-history-android

Yle. 2013. Suomen pelialan loistoajat jatkuvat - silti ”99 % hyvistä ideoista jää toteutmatta”. Viitattu 26.11.2013.
http://yle.fi/uutiset/suomen_pelialan_loistoajat_jatkuvat_-_silti_99__hyvista_ideoista_jaa_toteuttamatta/6487733

Youtube, 2013. Corona SDK: Physics in 5 lines. Viitattu 13.11.2013.
http://www.youtube.com/watch?v=AGSLDpqD_JM

Yxmantu, 2012. Marketing should be thrown away. Viitattu 3.12.2013.
http://yxmanty.tumblr.com/page/2

Zammetti, F. 2013. Learn Corona SDK Game Development. Kindle Locations 477–479. Viitattu 25.9.2013. Apress. Kindle Edition.)

Zammetti, F. 2013. Learn Corona SDK Game Development. Kindle Location 1569. Viitattu 26.11.2013. Apress. Kindle Edition.