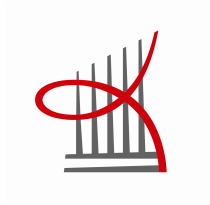


# Alustariippumattoman 2D-mobiilipelin prototyypin kehitys Cocos2d-x-pelimoottorilla

Henri Koskinen

Opinnäytetyö
Joulukuu 2014
Tietojenkäsittelyn koulutusohjelma
Ohjelmistotuotanto

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

**TIIVISTELMÄ**

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Ohjelmistotuotanto

KOSKINEN, HENRI:
Alustariippumattoman 2D-mobiilipelin prototyypin kehitys Cocos2d-x-pelimoottorilla

Opinnäytetyö 32 sivua
Joulukuu 2014

---

Tämän opinnäytetyön tavoitteena oli selvittää, kuinka eri mobiilialustat ja laitteet tulisi ottaa huomioon, kun kehitetään peliä Cocos2d-x-pelimoottorilla. Tarkoituksena oli kehittää Cocos2d-x-pelimoottorilla pieni mobiilipelin prototyyppi, jonka kehityksessä otettaisiin huomioon näiden eri alustojen ja laitteiden vaatimukset ja tutkittaisiin kuinka Cocos2d-x auttaa saavuttamaan tämän.

Cocos2d-x-pelimoottorilla pystyy luomaan melko pienellä panostuksella yksinkertaisen mobiilipelin, joka toimii iOS-, Android- ja Windows Phone -käyttöjärjestelmissä. Peliä ei kuitenkaan ole vielä julkaistu, sillä sen kehitystä jatketaan edelleen. Tavoitteena on, että peli julkaistaan vuoden 2015 alussa kaikille kolmelle mobiilialustalle.

Cocos2d-x-pelimoottori on hyvä valinta alustariippumattoman 2D-mobiilipelin luomiseen. Se ei välttämättä ole niin helppokäyttöinen kuin muut markkinoilla olevat vaihtoehdot, mutta sen ilmainen saatavuus, aktiivinen yhteisö ja avoin lähdekoodi ovat sellaisia etuja, jotka kannattaa pitää mielessä pelimoottoria valittaessa.

---

Asiasanat: Cocos2d-x, alustariippumattomuus, peliohjelmointi, mobiilipelit

**ABSTRACT**

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

KOSKINEN, HENRI:
Cross-platform 2D Mobile Game Prototype Development with Cocos2d-x Game Engine

Bachelor's thesis 32 pages
December 2014

---

The objective of this thesis was to find out how to take into account different mobile operating systems and devices when creating a game with Cocos2d-x game engine. The purpose was to develop a small mobile game prototype with Cocos2d-x while keeping in mind the limits and restrictions of multiple different platforms and devices.

Cocos2d-x helps developer to create mobile game for multiple platforms. It hides most of the platform dependent code from the developer to be able to focus on developing the game itself. The prototype will be released for multiple platforms at the beginning of year 2015.

Cocos2d-x game engine is a good choice when creating cross-platform 2D mobile games. It is not necessarily as easy to use as some other engines or frameworks, but it is free, active community and open source code are some of its benefits that should be kept in mind when choosing a game engine.

---

Keywords: Cocos2d-x, cross-platform, game programming, mobile games

**SISÄLLYS**

# LYHENTEET JA TERMIT

| | |
|---|---|
| Alustariippumattomuus | Ohjelmisto tai ohjelmointikieli, joka ei ole riippuvainen tietystä käyttöjärjestelmästä tai laitteistosta |
| Avoin lähdekoodi | Ohjelmiston lähdekoodi on vapaasti luettavissa, muokattavissa ja levitettävissä lisenssiehtojen mukaisesti |
| Cocos2d-x | Ilmainen avoimen lähdekoodin pelimoottori, jolla voi kehittää pelejä usealle eri alustalle |
| Pelimoottori | Ohjelmistokehys, jonka päälle kehittäjä voi rakentaa pelejä eri alustoille ja laitteille |
| Shoot ’em up | Ammuntapelien alalajityyppi, jossa tarkoituksena on ampua paljon ja väistellä vihollisten ammuksia |

# 1 JOHDANTO

Idean tälle opinnäytetyölle sain työharjoittelujaksoni lopulla, kun silloinen työharjoitteluyritykseni GameLayer ja minä mietimme olisiko syytä tutkia Unity- vai Cococs2d-x -pelimoottoria 2D-mobiilipelin toteutuksen kannalta. Päätimme, että keskittyisin Cocos2d-x:ään, sillä silloisen Unityn 2D-ominaisuudet eivät olleet sillä tasolla kuin ne tällä hetkellä ovat.

Keskittyminen 2D-mobiilipeleihin oli mielekäs valinta, sillä itselleni ne olivat kaikista tutuimpia, mutta myös siksi, että ne ovat yksi sovelluskauppojen suosituimmista alueista. Suomessa on myös useita mobiilipeleihin keskittyneitä yrityksiä, joille tämä työ voisi toimia eräänlaisena portfolioni osana työnhaussa.

Tämän opinnäytetyön tavoitteena on selvittää millaisia asioita tulee ottaa huomioon, kun tehdään alustariippumatonta 2D-mobiilipeliä. Tässä työssä ei siis käydä läpi 3D-grafiikkaan, konsoleihin tai PC-peleihin liittyviä asioita. Aihe on siis tarkasti rajattu 2D-mobiilipeleihin. Mobiilipeleillä tässä yhteydessä tarkoitetaan pelejä, jotka toimivat joko puhelimissa tai tableteissa. Aihetta käydään läpi iOS-, Android- ja Windows Phone-käyttöjärjestelmien kautta. On olemassa muitakin mobiilikäyttöjärjestelmiä, mutta nämä kolme ovat opinnäytetyön kirjoitushetkellä ne varteenotettavimmat. Mainitsen myös muiden käyttöjärjestelmien olemassaolosta, mutta niihin en syvällisemmin paneudu.

Alustariippumaton mobiilipelin kehitys -kappaleessa avaan alustariippumattomuutta ja tutustun olemassa oleviin mobiilikäyttöjärjestelmiin ja laitteisiin. Pelimoottorit ja ohjelmistokehykset -kappaleessa käyn läpi hieman syitä, miksi pelimoottorin tai ohjelmistokehyksen käyttäminen on hyvä idea. Vertailen myös muutamaa olemassa olevaa pelimoottoria. Haasteet mobiilipelin kehityksessä -kappaleessa esittelen opinnäytetyöni ongelman. Cocos2d-x-kappaleessa käyn läpi Cocos2d-x:n historiaa, mitä se tarjoaa kehittäjälle ja kuinka se helpottaa kehittäjän työtä. Cocos2d-x-prototyyppi -kappaleessa käyn läpi pienen mobiilipeli prototyypin kehittämistä, erityisesti Cocos2d-x:n näkökulmasta. Pohdinta-kappaleessa pohdin mitä opin, mitä ongelmia oli, mitä olisi voinut tehdä toisin tai paremmin. Viimeisessä jatkokehitys ja tulevaisuus -kappaleessa otan hieman kantaa ja kerron oman arvioni siitä kuinka projektia voisi jatkaa ja mitä se pitäisi sisällään.

# 2 ALUSTARIIPPUMATON MOBIILIPELIN KEHITYS

## 2.1 Alustariippumattomuus

Alustariippumattoman kehityksen tarkoituksena on piilottaa alustakohtainen koodi ja luoda erillinen taso tämän alustakohtaisen koodin päälle. Näin pyritään tarjoamaan kehittäjälle yksi yhteinen rajapinta, jolla kehittäjän on helppo kirjoittaa koodia, joka toimisi useilla laitteilla. Näin vältyttäisiin koodin uudelleenkirjoittamiselta, mikä on turhaa ajan ja resurssien tuhlaamista. Tarkoituksena on siis eräänlainen musta laatikko, joka kehittäjälle näkyy vain yhtenä rajapintana, mutta rajapinnan toteutusosa toteutetaan jokaiselle alustalle erikseen. Tällä tekniikalla pyritään kattamaan mahdollisimman paljon koodin kirjoittamisesta, mutta aina jää vähän alustakohtaista koodia, jota ei yksinkertaisesti voida, tai ei ole käytännöllistä piilottaa.

Ilman alustariippumattomia tekniikoita kehittäjien tulisi hallita kakkien haluttujen kohdealustojen ohjelmointikielet ja kehitysympäristöt, tai hankkia eri alustojen asiantuntijoita. Tämä usein lisää kehityskuluja, mutta myös ylläpitokustannukset kasvavat, kun koodista on olemassa jokaiselle alustalle omat versionsa. Ja jos markkinoille julkaistaan uusi käyttöjärjestelmä, on sille luotava ohjelmisto uudelleen lähes tyhjästä. Alustariippumattomien tekniikoiden käyttö onkin taloudellisempi vaihtoehto, sillä yleensä riittää, kun kehittäjä osaa tällaisen tekniikan käytön, näin ei tarvita jokaisen kohdealustan asiantuntijaa. Ylläpitokustannukset pienenevät myös, kun valtaosa koodista toimii sellaisenaan useimmilla valitun alustariippumattoman tekniikan tukemilla alustoilla.

Alustariippumattomuus on muutakin, kuin pelkkää ohjelmointia. Täytyy pyrkiä ottamaan huomioon myös alustojen muut piirteet. Peleissä tarvitaan paljon muitakin resursseja kuin vain lähdekoodia. Esimerkiksi kuva-, ääni- ja kirjasin -resurssit ovat sellaisia, mitkä voivat vaatia eri alustoilla eri tiedostomuodot, erilaiset tiedostopolun määrittelyt tai resurssitiedostojen kokojen suhteen on rajoituksia. Erilaisten sensorien ja käyttäjän syötteen vastaanottamisessakin saattaa olla eroja. Pitää pyrkiä mahdollisuuksien mukaan kulkemaan kultaista keskitietä, joten valinnat pitäisi tehdä sen mukaan mikä aiheuttaa vähiten lisätyötä ja mikä toimisi sellaisenaan tai pienillä muutoksilla useimmilla alustoilla.

## 2.2 Mobiilialustat

Mobiilikäyttöjärjestelmiä on olemassa hyvinkin monia, mutta kun puhutaan realistisista vaihtoehdoista niin silloin käyttöjärjestelmien määrä vähenee huomattavasti. Kirjoitushetkellä on olemassa oikeastaan vain kaksi todellista käyttöjärjestelmävaihtoehtoa, kun halutaan saavuttaa mahdollisimman laaja käyttäjäkunta kehitettävälle pelille tai sovellukselle.

Ensin tarkastellaan maailmanlaajuisesti markkinaosuutensa puolesta suosituinta Googlen Android-käyttöjärjestelmää, se on avoimen lähdekoodin alusta, jonka kehittämistä Google ohjaa. Androidin valtavaa suosiota selittää osaltaan se, että siitä voi kuka tahansa tehdä oman versionsa. Android tarjoaa laitteita myös sinne halvimpaan sektoriin, eikä vain kalliita lippulaivamalleja, näin esimerkiksi köyhemmissä maissa on varaa myös älypuhelimiin. Android on tullut paikkaamaan vanhojen Nokian Symbian -puhelinten jättämää aukkoja halvemmilla markkinoilla.

Toisena markkinaosuuksissa tulee Applen iOS. Apple ei ole julkaissut halpapuhelimia, vaan luottaa siihen, että sen laadukkaista tuotteista ollaan valmiita maksamaan, suuriakin summia. Applen iOS-käyttöjärjestelmä on suljettu ja toimii vain Applen omissa laitteissa. Tässä on kuitenkin puolensa, kuten myöhemmin huomataan. Käyttöjärjestelmälistauksessa kolmantena on Microsoftin Windows Phone muiden pienempien käyttöjärjestelmien kanssa (taulukko 1).

TAULUKKO 1. Asennettujen mobiilikäyttöjärjestelmien osuudet. (Tomi T. Ahonen 2014, muokattu)

| **Käyttöjärjestelmä** | **Kappaleet** | **Markkinaosuus** |
|---|---|---|
| Android | 1,336 M | 72 % |
| iOS | 359 M | 19 % |
| Windows Phone | 53 M | 3 % |
| Blackberry | 44 M | 2 % |
| Symbian | 41 M | 2 % |
| Muut | 10 M | 1 % |

Maailmanlaajuisesti Windows Phone -käyttöjärjestelmän asema ei kovin merkittävä ole, mutta kun otetaan tarkasteluun esimerkiksi Suomen tilanne, niin Windows Phone -laitteiden määrä ohittaakin iOS-käyttöjärjestelmää käyttävät laitteet suosiossaan (Pitkänene 2014).

Tähän tietoon pohjautuen päätin ottaa sen mukaan kolmantena käyttöjärjestelmänä. Uskoin myös siihen, että alustariippumattomia tekniikoita käyttäen Windows Phone -laitteiden mukaantulo ei työmäärääni suuresti kasvattaisi, joten katsoin, että se olisi mahdollisen lisätyön arvoista.

## 2.3 Mobiililaitteet

Mobiililaitteet on jaoteltu pääosin kahteen luokkaan, puhelimiin ja taulutietokoneisiin josta voidaan käyttää myös nimeä tabletti. Nykyisin voidaan nähdä käytettävän myös termia phablet, mutta itse en sitä tämän opinnäytetyön puitteissa käytä. Phablet-termillä tarkoittaa suurikokoisilla näytöllä varustettuja puhelimia ja käytännössä ne yhdistävät puhelinten ja tablettien ominaisuuksia. Termi phablet tulee phone- ja tablet-sanojen yhdistelmästä (Wikipedia 2014).

Tabletit ovat kannettavia tietokoneita huomattavasti pienempiä yksiosaisia kosketusnäyttölaitteita, joita ohjataan yleensä joko sormin tai tarkoitukseen varatulla kynällä. Nykyään on tarjolla jo ulkoisia näppäimistöjä, mitkä pyrkivät parantamaan käytettävyyttä erityisesti jos tarkoituksena on kirjoittaa paljon. Tabletteja on useita erikokoisia, mutta yleensä koot vaihtelevat seitsemän ja kymmenen tuuman välillä. Tosin löytyy markkinoilta yli kymmenen tuumaisiakin tabletteja. Tabletit harvoin ovat todellisia kehitystyökaluja itsessään, mutta toisinaan niitä voidaan käyttää ylläpidollisiin tehtäviin. Yleensä tabletteja kuitenkin käytetään lähinnä viihdetarkoituksiin. Tabletteja käytetään yleensä lukemiseen, internetin selaamiseen, musiikin kuunteluun ja videoiden katseluun. Eräs tärkeimmistä käyttötilanteista on juuri pelaaminen.

Puhelimet ovat kaikille tuttuja jokapäiväisiä arkiesineitä, jotka kulkevat mukana lähes jatkuvasti. Nykyään puhelimet ovat niin paljon kehittyneempiä ja tehokkaampia kuin muutamia vuosia sitten. Perinteisten tehtäviensä lisäksi, erilaiset sovellukset, pelit, internet ja sosiaalinen media ovat vieneet puhelimen käyttöä uuteen suuntaan. Enää se ei

ole laite, jolla vain soitetaan tai vastataan tekstiviesteihin, nykyään se on paljon muutakin ja se näkyy myös katukuvassa, sillä suuri osa ihmisistä kulkee pää kumarassa puhelimen näyttöä katsellen.

On arvioitu, että mobiilipelimarkkinat kasvavat vuoteen 2017 mennessä liikevaihdoltaan jopa 40 miljardiin dollariin, vaikka numerot eivät täysin paikkaansa pitäisikään, on suunta nousujohteinen (Pearson 2014).

# 3 PELIMOOTTORIT JA OHJELMISTOKEHYKSET

Pelimoottorit ja ohjelmistokehykset ovat olennainen osa nykypäivän pelinkehitystä, sillä harvoin projektia aletaan enää luomaan tyhjästä. Yleensä käytetään pelimoottorin tai ohjelmistokehyksen valmiita komponentteja, jotta päästään nopeammin itse pelinkehitykseen. Pelinkehityksessä oleellisinta on pelin varsinainen suunnittelu, pelin hahmot, sen tarina, pelattavuus, pelaajan hahmon ohjaaminen ja ylipäätään kaikki ne asiat mitkä tekee peleistä uniikkeja. Pelinkehityksessä ei niinkään keskitytä erilaisten teknologioiden kehittämiseen, vaan tarkoituksena on lähinnä valita se tekniikka, mikä parhaiten sopii halutunlaisen pelin kehittämiseen. Pelimoottorit siis antavat mahdollisuuden keskittyä itse pelinkehitykseen ja tämä antaa mahdollisuuden myös niille, joilla esimerkiksi on hyvä idea, mutta ei niin hyviä ohjelmointitaitoja tai aikaa paneutua alla oleviin teknologioihin.

Pelimoottorit ja ohjelmistokehykset helpottavat ja nopeuttavat kehittäjän työtä, koska usein ne tarjoavat alustariippumattoman yhteisen rajapinnan, jonka avulla voidaan kehittää peliä niin, että se toimii pelimoottorin tai ohjelmistokehyksen tukemilla alustoilla.

Pelimoottorilla tarkoitetaan yleensä valmiimpaa ennalta määrättyä pakettia, jossa asioita tehdään pelimoottorin haluamalla tavalla. Pelimoottorit ovat usein myös luotu erityyppisille pelityypeille. Esimerkiksi 2D-, 3D- tai räiskintä -peleille voi olla omat pelimoottorinsa, mitkä helpottavat erityisesti näiden pelityyppien ongelmien ratkaisemista. Muutamia suositumpia pelimoottoreita kirjoitushetkellä on Unity, Unreal Engine ja Cocos2d-x.

Ohjelmistokehykset ovat lähinnä työkaluja, jotka tarjoavat hyödyllisiä metodeita ja yhteisiä rajapintoja esimerkiksi peittämään useiden erilaisten peliohjainten eroja, jotta kehittäjä voi keskittyä pelinkehitykseen. Ohjelmistokehykset harvoin ottavat kantaa siihen, millaisia pelejä sillä tulisi tehdä tai mihin ne parhaiten sopisivat. Ne siis tarjoavat vapaamman ympäristön toimia ja antavat kehittäjälle enemmän vastuuta asioiden hoitamisesta.

Seuraavaksi pyrin esittelemään muutamia vaihtoehtoja pelimoottoreista ja ohjelmistokehyksistä. Käsiteltäväksi olen valinnut näennäisesti suosituimpia työkaluja, enkä käy

tässä läpi kaikkia olemassaolevia työkaluja läpi. Pelimoottoreita ja ohjelmistokehyksiä on kymmeniä, ellei jopa satoja ja hyvinkin marginaalisille käyttäjäryhmille.

Unity on varmasti alaa enemmän tai vähemmän seuranneille se tutuin. Tämä on varmasti yksi suosituimmista pelimoottoreista varsinkin lännessä ja se tarjoaa käyttäjälleen joko ilmaisen version tietyillä rajoitteilla tai sitten voit ostaa lisenssin, joka tarjoaa kaikki ominaisuudet kehittäjän käyttöön. Unityä käsittelevää materiaalia on olemassa paljon, huomattavasti enemmän kuin esimerkiksi Cocos2d-x käsittelevää. Unityllä voit julkaista pelejä useille alustoille, muun muassa kaikki selaimet ovat tuettuina, kuten myös Android, iOS, Windows Phone, Blackberry käyttöjärjestelmät, sekä useimmat konsolit. Unityn rinnalla toimii myös Asset Store. Se on kauppapaikka, josta kehittäjä voi ostaa valmiita 3D-malleja, 2D-grafiikkaa, projektipohjia, ääniä ja melkein mitä tahansa muutakin, mitä kehittämisessä saattaa tarvita. Unity on totuttu näkemään enemmän 3D-pelien luomistyökaluna, mutta se on myös parantanut asemiaan 2D-kehitystyökalujensa osalta. Päivityksessä 4.3 Unity toi viralliset 2D-pelien kehittämiseen tarkoitetut työkalut (Unity 2014).

HTML5 on yksi lupaavimmista alustariippumattomista tekniikoista. Se toimii laitteessa kuin laitteessa, missä vain on mahdollisuus internetselaimen käyttöön. Toinen tapa tuoda HTML5-tekniikalla luotu sovellus tai peli esimerkiksi puhelimeen on pakata se natiivin sovelluksen sisään. Näin sovellusta voidaan tarjota mobiililaitteiden omissa sovelluskaupoissa. Tämä ei kuitenkaan aina ole toimiva vaihtoehto, sillä itse kokeilin kehittää peliä PhoneGap-ohjelmistokehyksellä, mutta jouduin toteamaan, ettei ruudunpäivitys olisi hyväksyttävällä tasolla. Siitä tosin on aikaa ja tekniikka kehittyy jatkuvasti, joten tilanne voisi olla vuoden päästä jo aivan toinen.

Marmalade on yksi tarjolla olevista alustariippumattomista kehitysympäristöistä, joka on erityisesti keskittynyt mobiilialustoihin. Marmalade tukee iOS, Android, Windows Phone, Blackberry sekä Tizen alustoille julkaisemista, mutta Marmalade-sovellukset on mahdollista kääntää myös Windows- ja Mac -alustoille. Marmaladesta on tarjolla ilmainen rajoitettu versio ja erilaisia maksullisia lisenssivaihtoehtoja.

Marmaladesta on tarjolla myös Marmalade Quick -versio, jolla on mahdollista kehittää käyttäen Lua-kieltä. Mielenkiintoisen tästä tekee se, että Quick on rakennettu Marmaladen ja Cocos2d-x:n päälle (Marmalade 2014).

Eräs vaihtoehto on luoda oma pelimoottori, joka piilottaa eri mobiilialustojen laitteistokohtaiset piirteet. Tämä on se kaikista vaikein ja aikaa vaativin tie, sillä tällainen projekti kestää pitkään ja kehittäjä joutuu tekemään paljon työtä ennen kuin pääsee kehittämään varsinaista peliä. Tämä vaihtoehto opettaisi niin paljon kehittäjälleen, koska asioita joutuisi pohtimaan ja selvittämään pintaa syvemmältä. Pääsisi näkemään sen puolen, minkä markkinoilla jo olevat pelimoottorit ja ohjelmistokehykset piilottavat. Tämä vaihtoehto ei ole taloudellisesti kovin kannattava, sillä työmäärä on suuri ennen kuin omalla pelimoottorilla toteutettu peli pääsee markkinoille. Harvoin tilanne on myös sellainen, että omalla pelimoottorilla saataisiin merkittävä etulyöntiasema muihin markkinoilla oleviin tuotteisiin nähden, että se tekisi omasta pelimoottorista yhtään sen kannattavampaa.

# 4 HAASTEET MOBIILIPELIN KEHITYKSESSÄ

Mobiilipelien kehittäjien on valmistauduttava kohtaamaan erilaisia haasteita kehittäessään pelejä näin kirjavalle laitekannalle. Ehkä yksi suurimmista haasteista on kuitenkin laitteiden valtava määrä. Erityisesti Android-käyttöjärjestelmällä varustettuja laitteita on valtavasti. OpenSignalin tekemän tutkimuksen mukaan heidän sovellustaan latasi yli 18 000 eri Android-laitetta. Tämä kertoo osaltaan vastassa olevasta haasteesta, kun puhutaan esimerkiksi pelien testaamisesta Android-alustalla (OpenSignal, 2014). iOS-laitteita on tähän verrattuna olematon määrä. Tämä voi osittain selittää myös sen, miksi iOS on suosittu alusta kehittäjien keskuudessa. On helppoa testata ja kehittää peliä alustalle, jota tukevat laitteet voi ostaa melkein jokainen hieman suurempi yritys ja näin varmistua, että pelit toimivat kuten on haluttu.

Laitteiden suuri määrä tarkoittaa sitä, että kehitystyötä tehdään suurelle määrällä erilaisia näyttöjä, jotka ovat muun muassa fyysisesti eri kokoisia. Näytöt on varustettu erilaisilla resoluutioilla, kuvasuhteilla ja pikselitiheyksillä. Näistä suurin ongelma on kuvasuhteiden muuttuminen, sillä se saattaa aiheuttaa grafiikan vääristymistä mikäli sitä ei oteta huomioon.

Tämä asettaa omat haasteensa pelien graafisille vaatimuksille, sillä on toivottavaa, että peli näyttäisi mahdollisimman hyvältä laitteesta riippumatta. Oikotietä tälläisen saavuttamiseen ei kuitenkaan ole, sillä ei ole järkevää luoda grafiikkaa isoimmalle näytölle ja antaa pienempien laitteiden vain skaalata kuvat pienemmäksi. Tässä hukataan aivan turhaa muistia ja heikompitehoisten laitteiden resursseja, kun ne joutuvat käsittelemään isoja kuvia, vaikka eivät niistä mitään hyödy. Toinen tärkeä asia ottaa huomioon on kuvien sijoittelu näytölle. Kehittäjä ei voi kuvitella pelin näyttävän samalta eri laitteissa, jos graafiset elementit sijoitetaan näytölle absoluuttisilla pikselimäärityksillä.

Laitteiden välillä on myös muita eroja, kuin vain näyttöön tai sen kokoon liittyvät eroavaisuudet. iOS-, Android- ja Windows Phone -laitteet eroavat toisistaan fyysisten painikkeiden määrällä ja niiden toiminnoilla. Saatavilla olevien sensorien määrä vaihtelee laitteiden välillä. Kiihtyvyyssensori ja GPS ovat yleisiä, mutta sensorit, jotka mittaavat lämpötilaa tai kosteutta ovat puolestaan harvinaisia.

Kuinka siis pelimoottorit auttavat pelinkehittäjiä ratkaisemaan tällaiset ongelmat, tässä opinnäytetyössä tarkastelen Cocos2d-x:n vastausta tähän kysymykseen.

# 5 COCOS2D-X

Cocos2d-x on haarautunut Cocos2d-projektista omaksi kehityshaarakkeeksi kiinalaisen Zhe Wangin toimesta. Alunperin Cocos2d-projekti luotiin Python-kielellä, mutta alkuperäinen kehittäjä Ricardo Quesada kirjoitti Cocos2d:n uudelleen Objective-C:llä Applen AppStore:n perustamisen jälkeen, jotta Cocos2d:llä pystyi kehittämään nopeasti sovelluksia Applen alustalle. Tämä Cocos2d-projekti on toiminut pohjana useille eri Cocos2d-versioille (Wikipedia).

Cocos2d-x on avoimen lähdekoodin ohjelmisto. Avoin lähdekoodi tarkoittaa ohjelmistoa, jonka lähdekoodi on vapaasti tutkittavissa, korjattavissa ja paranneltavissa (Opensource.com). Cocos2d-x:n oma koodi löytyy suositusta GitHub-palvelusta, josta sitä voi halutessaan tutkia. Cocos2d-x on lisensoitu MIT-lisenssin alle. MIT-lisenssi on yksi lyhyimmistä ja vapaimmista lisensseistä, ja se sallii käyttäjälleen varsin suuren vapauden. MIT tulee sanoista Massachusetts Institute of Technology. MIT-lisensoidulla tuotteella voi tehdä melkein mitä vain, kunhan alkuperäiset tekijänoikeustiedot ja itse MIT-lisenssi pysyy (Cameron Chapman 2010). Cocos2d-x on myös täysin ilmainen, joten hankintakustannukset eivät nouse esteeksi pelimoottorin hankinnalle.

Cocos2d-x:llä voi kehittää joko C++-, Lua-, tai JavaScript -kielellä ja ohjelmistoja voi kehittää useille eri alustoille. Esimerkiksi iOS, Android, Windows Phone ja Blackberry ovat tuettuina. Zynga, Gamevil, Glu, Konami, Disney Mobile ovat muutamia tunnettuja yrityksiä, jotka ovat käyttäneet Cocos2d-x-pelimoottoria. Googlen, Microsoftin, AR-Min, Intelin ja Blackberryn insinöörit osallistuvat aktiivisesti Cocos2d-x:n kehitykseen. (Cocos2d-x).

Eräs Cocos2d-x:n vahvuus on ehdottomasti sen aktiivinen yhteisö. Useisiin ongelmiin löytyy vastaukset Cocos2d-x:n foorumilta ja IRC-kanavalta. Tämä on hyvä, sillä Cocos2d-x:n dokumentaatio ei ole niin hyvä, kuin se voisi olla. Cocos2d-x:ää koskevaa kirjallisuuttakin voisi olla enemmän.

Cocos2d-x-pelimoottori valittiin osaksi näistä syistä, mutta osaksi myös sen vahvojen 2D-ominaisuuksien ansiosta. Cocos2d-x tukee paljon muutakin kuin vain kuvien liikuttelua. Esimerkiksi Box2d- ja Chipmunk -fysiikkamoottorit ovat paljon käytettyjä Co-

cos2d-x:n yhteydessä. Cocos2d-x tarjoaa käyttäjälleen myös valmiita partikkeliefektejä, animaatioita ja kaikkea, mitä 2D-pelin kehityksessä tarvitsee. Kehittäjä voi kuitenkin luoda täysin omat ratkaisut kaikkiin haasteisiin, mikäli Cocos2d-x:n valmiit ratkaisut eivät miellytä.

Cocos2d-x on ollut myös kantavana voimana kahdessa suomalaisessa mobiilipelihitissä. Frogmindin Badland ja Fingersoftin Hill Climb Racing ovat rakennettu Cocos2d-x:llä käyttäen hyväksi juuri mainitsemaani Box2d-fysiikkamoottoria.

Tänä syksynä järjestetyssä kehittäjäkonferenssissa todettiin, että yli 70 prosenttia parhaista peleistä Kiinassa on tehty Cocos2d-x:llä. Konferenssissa puhuivat myös Googlen, Facebookin, Amazonin, Microsoftin, Intelin, ARMin ja Qualcommin edustajat, joten Cocos2d-x:llä on merkittäviä tukijoita (Cocos2d-x 2014).

Cocos2d-x:n opiskelua hidastaa sitä käsittelevien englanninkielisten kirjojen vähäinen määrä. Kirjoitushetkellä Amazon-kaupasta löytyy vain muutama sitä käsittelevä kirja. Onneksi internet tarjoaa muutamia hyviä sivustoja, jotka tarjoavat vinkkejä ja oppaita pelien toteuttamiseen Cocos2d-x:llä. Cocos2d-x:n mukana tuleva testiprojekti auttaa kuitenkin nopeasti alkuun ja ongelmiin löytyy usein ratkaisuja Cocos2d-x:n omalta foorumilta. Apua saattaa löytyä myös Cocos2d:n materiaaleista, sillä usein Cocos2d-x:n funktiot ovat vastaavia kuin Cocos2d:llä.

# 6 COCOS2D-X PELIPROTOTYYPPI

## 6.1 Idea

Idea tälle pelille syntyi todella helposti, sillä mielessäni on ollut jo pidempään Shoot ’em up -tyyppisen pelin tekeminen. Shoot ’em up -pelistä voidaan käyttää myös shooter-, STG- tai shmup -nimityksiä. Näillä nimillä kuvataan yleensä peliä joka on 2- tai 3-ulotteisella grafiikalla varustettua nopeatempoista avaruusräiskintää, jossa ruudulla edetään joko ylös tai oikealla, väistellään vihollisten ammuksia ja samalla pyritään tuhoamaan mahdollisimman paljon vihollisia (Racketboy 2012.).

Tämä tulisi olemaan myös oman pelini idea. Tässä luvussa kuvataan yksinkertaisen pelin oleellisimpia kehittämisvaiheita Cocos2d-x:n ja alustariippumattomuuden näkökulmasta. Mitä tulee esimerkiksi ottaa huomioon, kun asetetaan kuvia näytölle, valitaan äänitiedostoille oikeita tiedostomuotoja tai miten kuvat tulisi pakata, jotta saataisiin maksimaalinen hyöty, pieni muistinkulutus ja laadukas grafiikka?

## 6.2 Työkalut

Prototyyppiä luotiin Xcode- ja Visual Studio –kehitysympäristöissä. Nämä ovat iOS- ja Windows Phone -laitteiden viralliset kehitystyökalut. Androidille kehittäessä en käyttänyt Android-kehitykseen yleensä käytettävää Eclipse-editoria, vaan toimin suoraan komentorivikehoitteessa. Tämä siitä syystä, että Eclipselle ei todellista tarvetta ollut, sillä projektin kääntäminen Androidille onnistui ilman sitäkin varsin vaivattomasti.

Muita ohjelmistoja, mitä prototyypin valmistamisessa käytin, oli Adobe Illustrator, jolla loin oikeastaan kaikki pelin tarvitsemat grafiikat. Varsinaisia äänityökaluja minulla ei ollut, sillä pyrin löytämään taustamusiikit ja ääniefektit internetistä. Välillä jouduin muokkaamaan internetistä löytyviä äänitiedostoja, vaihtamaan tiedostomuotoa tai muuten editoimaan niitä, jotta saisin niistä parhaan hyödyn. Tässä tarkoituksessa käytin ilmaista Audacity-ohjelmistoa.

Aiemmin mainittujen lisäksi oleellisia työkaluja olivat TexturePacker, joka automatisoi kuvien pakkaamisen ja skaalaamisen, sekä Glyph Designer, jolla pystyin luomaan kuvatiedostoja haluamistani truetype-fonteista.

### 6.3 Ohjelmointi

Valtaosa prototyypin luomisesta oli ohjelmointia, joten tässä luvussa pyrin käymään läpi Cocos2d-x:n kanssa kehitettäessä tärkeimpien ominaisuuksien käyttöä. Läpikäytäviä asioita ovat muun muassa esikääntäjän komennot, Cocos2d-x:n muistinhallinta, Scene-elementtien hallinta, valikkojen rakentaminen ja muut tärkeänä pitämäni asiat.

Esikääntäjän komennot ovat C++-kielen mukana käytettävä ominaisuus. Cocos2d-x:llä kehittäessä tästä ominaisuudesta on hyötyä, jos jonkin koodin tarvitsee toimia eri tavalla eri alustoilla. Esimerkiksi tuki erilaisille äänitiedostomuodoille on erilainen eri alustojen kesken, joten tässä oli oiva tilaisuus käyttää ehdollista kääntämistä. Esikäännösvaiheessa selviää, mille alustalle ollaan kääntämässä ja tällöin myös selviää muuttujan arvo varsinaiseen käännösvaiheeseen. Alla olevasta esimerkistä voidaan nähdä miltä tälläinen esikäännösvaiheessa läpikäytävä koodi näyttää (kuvio 1).

```
// Music (WP8 does not support .mp3 files, so we have to use .wav files)
#if (CC_TARGET_PLATFORM == CC_PLATFORM_WP8) // .wav file
    const char* BG_MUSIC = "audio/music/495180_Space-Ninja-Squadro.wav";
#else                                       // .mp3 files for android and ios
    const char* BG_MUSIC = "audio/music/495180_Space-Ninja-Squadro.mp3";
#endif
```

KUVIO 1. Esikääntäjän ohjeita.

Cocos2d-x tarjoaa muistinhallintaan vastaavaa tapaa, kuin mikä on käytössä Objective-C-kielessä, se on saanut vaikutteita Objective-C:n automaattisesta viitteiden laskemisjärjestelmästä. Tämä johtuu siitä, että Cocos2d kirjoitetiin Objective-C:llä, ja Cocos2d-x on haarautunut kyseisestä projektista, joten se on ominut joitain tapoja luodakseen Cocos2d:n muistinhallinta järjestelmän uudelleen C++-kielellä. Suositeltava tapa on käyttää staattisia metodeita, tätä tapaa myös itse käytin (Engelbert 2013, 23).

Aloitan käymällä läpi projektin luomisen, jonka jälkeen keskityn kertomaan oleellisimmista luokista ja siitä, kuinka Cocos2d-x selvittää muun muassa erilaisten näyttökokojen hallinnan.

Aivan ensiksi tulee ladata haluttu Cocos2d-x:n versio heidän omilta kotisivuiltaan. Tässä käymme projektin läpi 2.5.5-versiolla. Käyttämässäni Cocos2d-x:n versiossa projekti luodaan yksinkertaisen Python-skriptin avulla. Siinä määritellään haluttu projektin nimi, pakkaus johon projekti halutaan ja lopuksi valitaan ohjelmointikieli. Voit valita yhden Cocos2d-x:n tukemista kehityskielistä, jotka ovat C++, Lua ja JavaScript. Joten esimerkiksi projekti voitaisiin luoda seuraavalla komennolla: python create_project.py –project DemoProjekti –package com.example.demoprojekti -language cpp. Tämä loisi minulle DemoProjekti-nimisen kansion, jossa kehityskieli olisi C++ ja joka olisi com.example.demoprojekti –pakkauksessa. Tämän jälkeen projekti voidaankin avata halutussa kehitysympäristössä ja rakentaa ensimmäisen kerran.

Edellä mainittujen alkutoimien jälkeen työ on ollut sitä perinteistä kehitystyötä, eli dokumentaation ja esimerkkien soveltamista ja kokeilemista. Itselleni suurin apu oli ehdottomasti Roger Engelbertin Cocos2d-x by Example Beginner’s Guide -kirja.

Ohjelmointiprosessin aikana pyrin siihen, että minulla oli koko ajan toimiva versio käännettävissä mille tahansa laitteelle. Tämän ansiosta minun oli helppo seurata kehityksen edetessä sitä, jos jossain kohtaa peliin olisi tullut ohjelmointivirhe, minkä takia peliä ei pystyisikään enää kääntämään onnistuneesti. Kääntämisellä tarkoitetaan ohjelmoinnissa sitä, että ohjelmoija kirjoittaa ihmisten ymmärtämää ohjelmakoodia, joka käännetään alustakohtaiseen muotoon, jotta laitteet voivat sitten ajaa eli suorittaa tekemäsi ohjelman.

Eräs ohjelmoinnissa, kehittämisessä ja erityisesti sen testaamisessa apuna ollut ohjelmointitapa, mitä käytin, oli Constants-otsikkotiedosto. Se sisälsi suuren määrän erilaisia vakiomuuttujia. Näiden muuttujien arvot olin poiminut lähdekooditiedostoista ja vaihtanut ne selkeämmillä Constants-tiedoston vakiomuuttujilla. Näin keskitetysti ja nimettyinä ne helpottavat koodin luettavuutta ja päivittämistä. Esimerkiksi fonttitiedoston nimen määrittäminen vaihtelee alustasta riippuen. iOS-laitteet lukevat fontit eri tavalla kuin vaikkapa Windows Phone -laitteet. Tästä syystä kirjoitinkin useita sääntöjä esikääntäjälle, mikä voisi ennen koodin varsinaista kääntämistä pitää huolen siitä, että haluamallani muuttujalla on sellainen arvo, mitä alusta tukee. Alla esimerkki iOS- ja Android -laitteiden erosta määritellä fonttien polut (kuvio 2).

```
// Fonts
#if (CC_TARGET_PLATFORM == CC_PLATFORM_IOS)
    const char* MENU_FONT = "Orbitron-Black";
    const char* HUD_FONT = "Tektrron-Regular";
    const char* CREDITS_FONT = "Orbitron-Black";
#else
    const char* MENU_FONT = "fonts/Orbitron-Black.ttf";
    const char* HUD_FONT = "fonts/Tektrron-Regular.ttf";
    const char* CREDITS_FONT = "fonts/Orbitron Black.ttf";
#endif
```

KUVIO 2. Constants.h tiedoston esikääntäjän ohjeita.

iOS-alustalla joutuu tekemään myös toisen työvaiheen, että saa fontit toimimaan. Joudut määrittelemään iOS-projektitiedostossa käyttämäsi fontit erikseen. Tätä jouduin miettimään hieman pidempään, että nimi mikä tähän kelpaa tulee hakea FontBook-ohjelmasta, joka näyttää fontin PostScript-nimen (kuvio 3).

| Fonts provided by application | Array | (4 items) |
|---|---|---|
| Item 0 | String | fonts/Orbitron-Bold.ttf |
| Item 1 | String | fonts/Orbitron-Medium.ttf |
| Item 2 | String | fonts/Orbitron-Light.ttf |
| Item 3 | String | fonts/Orbitron-Black.ttf |

KUVIO 3. iOS-projektitiedoston fonttimäärittelyt.

Eräs Cocos2d-x:n tärkeimmistä luokista on CCDirector. CCDirector hallitsee CCScene-objekteja ja tietää kaiken sovelluksestasi, esimerkiksi näytön koon, ruudunpäivitysnopeuden ja aktiivisena olevan CCScene-objektin (Engelbert 2013, 19).

Toinen tärkeä osa kokonaisuuden kannalta on CCScene-luokka, sen avulla peli voidaan jakaa loogisiksi kokonaisuuksiksi. CCScene-objekteja voivat olla esimerkiksi pelin päävalikko, asetusruutu ja itse pelivaihe. CCScene-objekteista syntyy pino, mikä toimii sillä periaatteella, että ensimmäisenä laitettu CCScene tässä pinossa poistuu siitä viimeisenä. Näiden välillä liikutaan CCDirector-luokan metodien avulla, joka pystyy laittamaan CCSenejä tähän pinoon, poistamaan niitä ja täten hallitsemaan näin pelin kulkua. Näiden siirtymisten välillä voi olla yksinkertaisia animaatioita, joita CCTransition-luokka tarjoaa. Voit esimerkiksi häivyttää olemassa olevan CCScenen mustaan, vaihtaa uuteen CCScene-objektiin ja liukua mustasta jälleen esille. CCScene-objekteihin voi liittää erinäisiä asioita. Yleensä niihin kiinnitetään CCLayer-luokan objekteja, jotka tarjoavat muun muassa käyttäjän kosketusten käsittelyä. Tähän CCLayer-objektiin voidaan liittää erilaisia CCSprite-luokan olioita, jotka periaatteessa ovat vain tavallisia kuvia, joita käyttäjä näkee laitteensa ruudulla. Näiden kolmen CCScenne-, CCLayer- ja

CCSprite-luokkien olioita järjestämällä voidaan saada aikaiseksi jo erittäin mukavia pelejä. Itse en omassa pelissäni juuri muita käyttänyt. 2D-pelit ovat vain kuvien liikuttamista ja järjestelyä, oli se sitten yksinkertainen kuva, partikkeliefekti tai animaatio, ne ovat vain kuvia.

Cocos2d-x tarjoaa helppokäyttöisen mahdollisuuden tallentaa erilaista tietoa käyttäjän laitteen muistiin, tässä käytetään CCUserDefault-luokkaa hyväksi. Esimerkkinä voisi olla vaikka musiikin ja ääniefektien voimakkuuden tallentaminen. Tallentaminen tapahtuu asetusruudulta poistuttaessa, jonka jälkeen ne pysyvät laitteen muistissa niin kauan kuin käyttäjällä on peli asennettuna. Kun peliä käynnistetään uudestaan, kyseiset asetukset luetaan ja pelin asetukset säädetään niin kuten käyttäjä oli ne säätänyt edellisellä kerralla, ilman että käyttäjän tarvitsee niitä uudelleen asettaa.

Yksinkertaisen valikkorakenteen luonti on Cocos2d-x:ssä todella helppoa. Alla olevasta kuvasta selviää pelissä käytössä oleva päävalikon rakenne (kuvio 3).

```
CCMenuItemFont* continueBtn = CCMenuItemFont::create("CONTINUE", this, menu_selector(MainMenuScene::continueBtnCallback));
CCMenuItemFont* playBtn = CCMenuItemFont::create("NEW GAME", this, menu_selector(MainMenuScene::playBtnCallback));
CCMenuItemFont* settingsBtn = CCMenuItemFont::create("SETTINGS", this, menu_selector(MainMenuScene::settingsBtnCallback));
CCMenuItemFont* creditsBtn = CCMenuItemFont::create("CREDITS", this, menu_selector(MainMenuScene::creditsBtnCallback));
CCMenuItemFont* quitBtn = CCMenuItemFont::create("QUIT", this, menu_selector(MainMenuScene::quitBtnCallback));

CCMenu* menu = CCMenu::create(continueBtn, playBtn, settingsBtn, creditsBtn, quitBtn, NULL);
menu->setPosition(ccp(visibleSize.width * 0.7f, visibleSize.height * 0.5f));
menu->alignItemsVerticallyWithPadding(visibleSize.height * 0.1f);
this->addChild(menu, 1);
```

KUVIO 3. Valikkorakenteen luominen.

Erityisen oleellista mielestäni on tarkastella sitä koodia, mikä keskittyy valitsemaan sopivan kuvakansion sd-, hd- ja hdr -kansioiden väliltä. Kyseisten kansioiden nimet kuvaavat erikokoisia kuvatiedostoja, joita on tarkoitus käyttää sen mukaan, minkä kokoinen kohdelaite on kyseessä. Alla olevasta lähdekoodista näemme, kuinka vertaamme laitteen näytön kokoa ennalta määrättyihin arvoihin, joiden perusteella valitsemme oikean kuvakansion (kuvio 4).

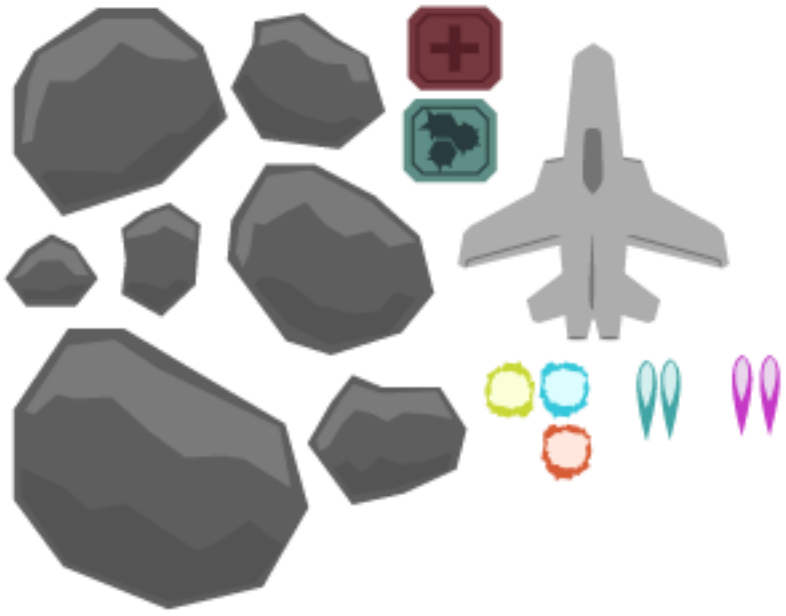
```
CCSize designSize = CCSizeMake(800, 480);
CCLog("Design size W:%.2f, H:%.2f", designSize.width, designSize.height);

eglView->setDesignResolutionSize(designSize.width, designSize.height, kResolutionFixedHeight);

CCSize frameSize = eglView->getFrameSize(); // get the real screen size in pixels
CCLog("Device screen size W:%.2f, H:%.2f", frameSize.width, frameSize.height);

// Set up different resource folders for different screens
std::vector<std::string> searchPaths;
if(frameSize.height >= 1080)
{
    searchPaths.push_back("hdr");
    director->setContentScaleFactor(frameSize.height / designSize.height);
}
else if(frameSize.height >= 720)
{
    searchPaths.push_back("hd");
    director->setContentScaleFactor(frameSize.height / designSize.height);
}
else
{
    searchPaths.push_back("sd");
    director->setContentScaleFactor(frameSize.height / designSize.height);
}
CCLog("The Content of the searchPaths is %s.", searchPaths[0].c_str());
CCFileUtils::sharedFileUtils()->setSearchPaths(searchPaths);

CCSize windowSize = director->getWinSize();
CCLog("Window Size W: %.2f, H: %.2f", windowSize.width, windowSize.height);


CCSize visibleSize = director->getVisibleSize();
CCLog("Visible Size W: %.2f, H: %.2f", visibleSize.width, visibleSize.height);
```

KUVIO 4. AppDelegate.cpp tiedoston koodi, joka valitsee oikean kuvakansion.

### 6.4 Grafiikka

Pelin grafiikka luotiin lähes yksinomaan Adoben Illustrator -ohjelmistolla. Muutamia hahmotelmia tehtiin myös Adoben Photoshop -ohjelmistolla. Grafiikan teossa keskityin tässä tapauksessa sen selkeyteen ja helppoon ymmärrettävyyteen pieneltäkin näytöltä. En siis halunnut tehdä liian yksityiskohtaista, sillä aikaa ei kovin paljoa ollut. Grafiikan teossa pyrin myös siihen, että pelillä olisi selkeä graafinen ilme, jonka kaikki elementit sopisivat yhteen toistensa kanssa. Grafiikan teossa tulisi myös muistaa se, että kuvat luotaisiin mahdollisimman suuriksi alussa. Tästä on se etu, että jos myöhemmin ilmenee tarvetta luoda esimerkiksi mainosmateriaaleja, julisteita tai muita suuria dokumentteja, kuvien laatu pysyisi hyvänä. Illustratoria käyttäessä sillä kuitenkaan ei ole niin suurta merkitystä, sillä se on vektorigrafiikkaohjelma, joten se takaa rajattoman skaalaamisen laadun heikentymättä. Vektorigrafiikka tarkoittaa grafiikkaa, joka ei muodostu pikseleistä vaan matematiikasta. Tämä ei kuitenkaan estä sitä, että kun vektorikuvat ovat tallennettu esimerkiksi .png-muodossa, ettei niiden laatu kärsisi, jos niitä aletaan skaalaamaan. Sillä ne ovat siinä vaiheessa tavallisia pikseleistä muodostuneita kuvia.

Seuraavaksi kun olin saanut haluamani grafiikat tehtyä Illustratorissa, vaihdoin Texture Packer –ohjelman pariin. Sillä pystyin helposti luomaan erikokoiset sprite sheet -kuvat. Sprite sheet –kuva tarkoittaa yksittäistä suurempaa kuvaa, joka on rakennettu useasta pienestä kuvasta. Tämän etuina on esimerkiksi muistin vähäisempi kulutus ja se, että pelin grafiikkaa piirtäessä se on paljon suorituskykyisempää kuin se, että joutuisi lataamaan ja piirtämään jokaisen kuvan erikseen. Näin voidaan käyttää yhtä ja samaa kuvaa, mikä vapauttaa resursseja muuhun käyttöön. TexturePackerissa on myös se etu, että sillä on helppo luoda erikokoiset kuvat oikeisiin sd-, hd- ja hdr -kansioihin, jota erikokoiset näytöt osaavat käyttää. Ei ole mitään järkeä luoda grafiikkaa esimerkiksi iPad-laitteen resoluutiolle, mikä on 2048 x 1536 ja käyttää sitä laitteessa jonka resoluutio on 800 x 480. Tämä on huono tapa toimia, sillä voidaan olettaa, että näin pienellä näytöllä varustettu laite ei muutenkaan kovin tehokas ole. Pelien suorituskyky kärsii liian suurien kuvien käsittelystä. Onkin huomattavasti parempi antaa TexturePackerin skaalata alkuperäistä suurempaa kuvaa sopivassa suhteessa, jotta saadaan kuva, joka paremmin käy laitteille, joissa on pienet näytöt. Cocos2d-x:n CCSpriteBatch-luokka osaa käyttää hyväkseen näitä Sprite sheet -kuvia. Tästä on se etu, että taustalla oleva grafiikkamoottori OpenGL ES tai DirectX Windows Phonen tapauksessa, pystyy piirtämään kaikki saman sprite sheet -kuvan elementit yhdellä kertaa. Vältytään siis kuvalähteen vaihtamiselta, mikä vie suorituskykyä. Tämä on eräs keino optimoida pelin suorituskykyä ja onkin todella yleinen ja käytetty tekniikka myös muissa pelimoottoreissa (kuvio 5).



KUVIO 5. Sprite sheet –muodossa oleva peligrafiikkatiedosto.

Tässä vaiheessa käytössä oli myös Glyph Designer -työkalu, jolla pystyin luomaan haluamistani fonteista kuvia, jotka toimivat sprite sheet -tekniikan tapaan. On syytä pitää mielessä, että Cocos2d-x tarjoaa mahdollisuuden myös käyttää truetype-fontteja sellaisinaan, se on toimiva vaihtoehto silloin, kun luodaan staattisia tekstejä. Staattisilla teksteillä tarkoitetaan tekstejä joiden sisältöä ei tarvitse muuttaa niiden luonnin jälkeen. Tämä ei kuitenkaan toimi esimerkiksi pistelaskurin kanssa, jonka tulee päivittyä jopa useita kertoja sekunnissa. On hyvä huomata, että vaikka truetype-fontin käyttö toisikin tiettyjä etuja bittikartta-kuvaan nähden, ongelma syntyy siinä, että jokainen kerta kun truetype-fontilla kirjoitettu teksti muuttuu, Cocos2d-x joutuu tekemään pinnan alla siitä uuden kuvan. Tässä on se ongelman ydin pistelaskurin ja truetype-fontin yhteydessä. Aina kun pistelaskurin arvo muuttuu, luodaan uusi teksti truetype-fontilla, siitä luodaan kuva ja se näytetään käyttäjälle päivittyneenä pistetilanteena. Sprite sheet-kuvia käytettäessä tilanne on kuitenkin toinen, numerot ovat valmiiksi kuvia ja ladattu muistiin, joten pelimoottorin ei tarvitse kuin piirtää oikea kohta tästä sprite sheet-kuvasta. On siis selkeästi parempi käyttää tälläisiä bittikarttakuvia. Tästä on se haitta, että jos haluat tekstin erivärisenä, erikokoisena tai vaikkapa suuraakkosin, joudut aina tekemään uuden kuvan näitä varten. Se on kuitenkin pieni hinta saavutetusta suorituskyvystä. Glyph Designer -ohjelma helpottaa näiden kuvien tuottamista huomattavasti. Tulee myös huomata, että nämä Glyph Designerin tuottamat kuvat tulee tarjota kaikille resoluutiotyypeille, joita pelissä aiotaan tarjota. Näin vältytään kuvien heikolta laadulta ja pelin laatu pysyy korkealla. Ei siis kannata oikoa, vaan tehdä asiat kunnolla. Alla esimerkki Glyph Designerin luomasta tekstitiedostosta, joka sisältää kaiken tarvittavan tiedon halutun merkin löytämiseen bittikartta-kuvasta, joka on liitetty myös kuvaan (kuvio 6).

```
info face="Orbitron" size=110 bold=0 italic=0 charset="" unicode=0 stretchH=100 smooth=1 aa=1 padding=0,0,0,0 spacing=2,2
common lineHeight=110 base=82 scaleW=512 scaleH=256 pages=1 packed=0
page id=0 file="game_text.png"
chars count=12
char id=32     x=325   y=86    width=0     height=0     xoffset=0     yoffset=82    xadvance=32    page=0 chnl=0 letter="space"
char id=43     x=277   y=86    width=46    height=46    xoffset=2     yoffset=26    xadvance=48    page=0 chnl=0 letter="+"
char id=48     x=2     y=2     width=82    height=82    xoffset=6     yoffset=0     xadvance=92    page=0 chnl=0 letter="0"
char id=49     x=236   y=86    width=39    height=82    xoffset=0     yoffset=0     xadvance=43    page=0 chnl=0 letter="1"
char id=50     x=86    y=2     width=82    height=82    xoffset=6     yoffset=0     xadvance=91    page=0 chnl=0 letter="2"
char id=51     x=170   y=2     width=82    height=82    xoffset=6     yoffset=0     xadvance=91    page=0 chnl=0 letter="3"
char id=52     x=86    y=86    width=78    height=82    xoffset=1     yoffset=0     xadvance=80    page=0 chnl=0 letter="4"
char id=53     x=254   y=2     width=82    height=82    xoffset=6     yoffset=0     xadvance=91    page=0 chnl=0 letter="5"
char id=54     x=338   y=2     width=82    height=82    xoffset=6     yoffset=0     xadvance=90    page=0 chnl=0 letter="6"
char id=55     x=166   y=86    width=68    height=82    xoffset=0     yoffset=0     xadvance=73    page=0 chnl=0 letter="7"
char id=56     x=422   y=2     width=82    height=82    xoffset=6     yoffset=0     xadvance=92    page=0 chnl=0 letter="8"
char id=57     x=2     y=86    width=82    height=82    xoffset=6     yoffset=0     xadvance=91    page=0 chnl=0 letter="9"
kernings count=0
```

KUVIO 6. Glyph Designerin tuottama tekstitiedosto ja Sprite sheet –kuva merkeistä.

## 6.5 Äänet

Äänien luomisprosessia en pysty kovin tarkasti kuvaamaan, sillä en tuottanut itse ääniefektejä tai musiikkia tähän projektiin. Tästä syystä jouduinkin turvautumaan internetin tarjoamiin lähteisiin. Se tarjosikin erinomaisia lähteitä ilmaisen ja vapaasti käytettävän musiikin löytämiseen. Yleensä tällaiset sivustot, jotka tarjoavat ilmaisia ääniefektejä tai taustamusiikkeja, eivät toivo muuta kuin mainintaa projektin tekijätiedoissa ja sitä, että alkuperäisten äänien lisenssit pysyvät. Tämä onkin melko pieni vaatimus ilmaisesta ja vapaasti käytettävästä musiikista, joka toisinaan saattaa olla hyvinkin laadukasta.

Käynkin tässä kohtaa läpi Cocos2d-x:n tukemia tiedostomuotoja ja hyvistä toimintatavoista Cocos2d-x:n kanssa.

Cocos2d-x tukee useiden ääniefektien samanaikaista toistoa. Ääniefektit ovat lyhyitä, muutaman sekunnin pituisia äänitiedostoja. Näistä esimerkkejä voisi olla räjähdys tai kolikon kerääminen. Cocos2d-x tukee myös pidempien taustamusiikkien soittamista, sillä rajoituksella tosin, että samanaikaisesti ei voi soida kuin yksi taustamusiikki. Tämä rajoitus jäi itseäni hieman harmittamaan, sillä olisin halunnut tehdä liukuvia, taustamusiikista toiseen vaihtuvia siirtymiä.

Cocos2d-x tukee useimpia suosituimpia äänitiedostomuotoja. Cocos2d-x:n mukana tuleva äänimoottori on kuitenkin hyvin alkeellinen, joten se ei osaa kuin perustoiminnot äänen soittamiselle, pysäyttämiselle, äänenvoimakkuuden säätämiselle ja muutamalle muulle perustoiminnolle. Cocos2d-x:n äänimoottori ei esimerkiksi Windows Phone -alustalla osaa toistaa kuin .wav- ja .mid-tiedostoja. Windows Phone -alustalla olisi mahdollista muitakin tiedostomuotoja toistaa, mutta se vaatisi natiiviin ohjelmointiin paneutumista, minkä koin projektin puitteissa liian vaativaksi. Tuki on melko vaihteleva ja tarkoittaa sitä, että joudutaan tarjoamaan samat tiedostot useassa muodossa.

Alla olevaan taulukkoon olen koonnut eri alustojen tiedostomuotoja (taulukko 2). Windows Phone -laitteet tarvitsevat .wav-muotoisia tiedostoja, tämä on erityisen ongelmallista sillä .wav on pakkaamaton tiedostomuoto. Tämä saattaa kasvattaa pelin kokoa

huomattavasti useilla megatavuilla. Tätä ongelmaa voi pyrkiä lievittämään laskemalla äänenlaadullisia asetuksia, muuttamaan stereo-äänen mono-ääneksi, eli tällöin äänestä poistetaan toisen äänikanavan tieto ja näin saadaan tiedostokokoa pienemmäksi. Tällöin äänen laatu kuitenkin aina kärsii, mutta toisinaan se on hyväksyttävä. Äänen suhteen joudutaan tekemään usein kompromisseja äänenlaadun ja tiedostokoon suhteen.

TAULUKKO 2. Eri tiedostomuodot. (Cocos2d-x, 2014 )

| **Alusta** | **Taustamusiikkimuoto** | **Ääniefektimuoto** |
|---|---|---|
| Android | .mp3 | .ogg |
| iOS | .mp3 tai .caf | .caf |
| Windows Phone | .mid tai .wav | .mid tai .wav |

Äänitiedostojen käytön suhteen on suositeltavaa, että ne ladataan valmiiksi muistiin ennen ensimmäistä kertaa, kun niitä tarvitaan. Tällä vältytään viiveeltä, mikä aiheutuisi siitä, kun äänitiedostot pitäisi ensin ladata, ennen kuin ne voisi soittaa. Tämä vaihe on hyvä suorittaa mahdollisimman aikaisessa kohdassa tai muuten sopivassa kohdassa ohjelmakoodia. Esimerkiksi itse latasin pelin taustamusiikin ja muutaman ääniefektin AppDelegate-luokassa, mikä on yksi ensimmäisiä luokkia, mitä Cocos2d-x lukee kun se käynnistää ohjelmaa.

Taustamusiikin kohdalla sain huomata sen kuinka suuria .wav-tiedostot ovatkaan. Muutaman megatavun .mp3-muotoisesta tiedostosta kasvoi yli 10 megatavun kokoinen. Ei ole hyvä, että yksittäinen tiedosto vie näin paljon tilaa, joten jouduin Audacity-ohjelmassa käsittelemään musiikin laadullisia asetuksia. Laskin äänen taajuutta 44100 hertsistä 11025 hertsiin. Se ei kuitenkaan vielä riittänyt, joten jouduin tekemään alkuperäisestä stereo-äänestä mono-äänen. Tämä tarkoittaa sitä, että toinen kanava poistuu äänitiedostosta kokonaan ja näin ollen myös äänitiedoston koko puolittuu. Lopulta olin päässyt melko säädylliseen parin megatavun tiedostokokoon, joka oli mielestäni ihan sopiva.

### 6.6 Testaus

Peliprojektia testattiin puhelimilla ja tableteilla, onneksi niitä löytyi myös jokaisella käyttöjärjestelmällä. Näin saatiin edes pientä suuntaa siitä, kuinka peli toimii eri laitteil-

la. Laitteet, joilla peliä testattiin, on esitelty alla olevasta taulukossa. Alimpana taulukossa oleva HTC Desire -malli edustaa ehdottomasti taulukon vanhinta ja heikointa laitetta, eikä sillä testaamista pidetty kovin oleellisena, mutta se on taulukkoon kuitenkin lisätty, koska testaamista tehtiin myös sillä (taulukko 3).

TAULUKKO 3. Henkilökohtaiset testilaitteet.

| **Malli** | **Käyttöjärjestelmä** | **Näytön resoluutio** | **Näytön koko** |
|---|---|---|---|
| LG Nexus 5 | Android 4.4.4 | 1920 x 1080 | 4.95 tuumaa |
| Asus Nexus 7 (1st) | Android 4.4.4 | 1280 x 800 | 7.0 tuumaa |
| Apple iPad Air | iOS 8.0.2 | 2048 x 1536 | 9.7 tuumaa |
| Nokia Lumia 820 | Windows Phone 8.1 | 800 x 480 | 4.3 tuumaa |
| HTC Desire | Android 2.3.3 | 800 x 480 | 3.7 tuumaa |

Vaikka taulukko sisältää puhelimia ja tabletteja kaikista käyttöjärjestelmistä, ei siihen tulisi liikaa luottaa. Esimerkiksi pelkästään erilaisia Android-laitteita on tuhansia, muiden alustojen kohdalla laitteiden määrä on huomattavasti pienempi. Tästä syystä erilaiset beta-versioiden testaukset ovat yleistyneet. Tämä tarkoittaa sitä, että ennen varsinaista julkaisua, peli julkaistaan rajatulle käyttäjäkunnalle testaukseen. Näin pelin toiminnasta saadaan huomattavasti kattavampi kuva, kun testaajien ja testilaitteiden määrä kasvaa.

### 6.7 Julkaisu

Julkaisua ei tämän opinnäytetyön puitteissa ehditty tekemään, eikä sitä käydä tämän syvällisemmin tässä opinnäytetyössä läpi. Julkaisu lykkääntyi pääosin siitä syystä, että tekijän halu luoda laajempi peli, kasvoi. Peliä tehdessä alkoi käydä selväksi, että tästä voisi saada hyvän pelin, jota voisi käyttää oman osaamisena näyttönä töitä hakiessa. Tästä syystä pelistä halutaan julkaista loppuun asti hiottu versio kaikille alustoille. Haluan keskittyä luomaan sisältöä peliin, jotta pelaajat voisivat viihtyä pitkään sen parissa. Näillä näkyminen julkaisu siirtynee vuoden 2015 alkuun.

# 7 POHDINTA

Tämän opinnäytetyön tavoitteena oli selvittää millaisia asioita kehittäjän tulisi ottaa huomioon kun kehitetään 2D-mobiilipeliä usealle eri mobiilikäyttöjärjestelmälle. Tätä tavoitetta selvitettiin erityisesti Cocos2d-x-pelimoottorin kautta. Toisena tavoitteena oli tutkia kuinka tällainen pelimoottori auttaa piilottamaan käyttöjärjestelmäkohtaisia asioita niin, että kehittäjä voisi keskittyä tekemään itse peliä. Se ei kuitenkaan poista sitä ongelmaa, että peli tulee suunnitella niin, että se toimii useilla laitteilla ja erilaisilla näytöillä. On hyvä pitää mielessä myös se, että kaikissa laitteissa ei välttämättä ole samoja sensoreita, tästä esimerkkinä GPS, kiihtyvyyssensori tai NFC, nämä asiat tulee ottaa huomioon kun peliä suunnitellaan. Käyttäjän syötteen vastaanottaminenkin saattaa erota alustojen välillä.

Cocos2d-x sopii mielestäni erinomaisesti pienten ja vähän isompienkin 2D-mobiilipelien toteutukseen. Se tarjoaa suuren määrän erilaisia kehitystä helpottavia ominaisuuksia. Cocos2d-x piilottaa hyvin laitekohtaiset yksityiskohdat, mutta joskus tulee kuitenkin tilanteita, joissa kehittäjä joutuu kirjoittamaan alustakohtaista koodia. Tämä oli mielestäni kuitenkin todella vähäistä ja onnistuu helposti esimerkiksi C++:n esikääntäjän komennoilla. Useat kolmannen osapuolen ohjelmistot tukevat myös Cocos2d-x:ää, esimerkiksi sellaisia ovat TexturePacker ja Glyph Designer. Näihin kannattaa muutenkin paneutua, sillä ne ovat todella hyödyllisiä ohjelmistoja. Cocos2d-x:n uudempi versio tuo mukanaan paljon uudistuksia ja parannuksia edelliseen Cocos2d-x pelimoottoriversioon, joten suosittelen siihen tutustumista vaikkei sitä käyty läpi tässä opinnäytetyössä.

Opinnäytetyöprosessin aikana sain paljon uutta tietoa, erityisesti siitä kuinka erikokoiset näytöt tulee ottaa huomioon, kun pelissä esimerkiksi asetellaan kuvia näytölle tai kun optimoidaan oikeankokoisia ja laatuisia grafiikoita näytölle. Tulin myös huomanneeksi, että näytön resoluutio tai fyysinen koko eivät ole ainoita asioita, mitä tulee ottaa huomioon kun peliä usealle alustalle kehitetään.

Opinnäytetyötä ei voi kuitenkaan täysin ongelmattomana pitää, sillä projekti ei pysynyt aikataulussa. Tästä tulee kuitenkin ottaa opiksi ja jatkaa seuraavan projektin pariin.

# 8 TULEVAISUUS JA JATKOKEHITYS

Tulevaisuutta ajatellen projekti jätti useita mielenkiintoisia kysymyksiä ja jatkokehittämisen aiheita, mutta tämä on mielestäni erinomainen asia. Sillä nyt, kun vahva perusta on luotu, on sen päälle hyvä rakentaa lisää entistä täsmällisempää tietoa ja taitoa. Erityisesti jäi halu paneutua sellaisiin asioihin kuten Free to Play -ansaintamalli, missä pelien sisäiset pienet ostokset ovat oleellisessa osassa. Free to play -mallilla tarkoitetaan sitä, että pelin saa itselleen ilmaiseksi, mutta sitä pelatessa voidaan ostaa esimerkiksi timantteja, joilla pelissä etenee nopeammin tai saa avattua uusia ratoja, ajoneuvoja, aseita, voimia tai mitä vain voisi keksiä.

Eräs tärkeä piirre mielestäni nykyisissä peleissä tuntuu olevan sosiaalinen media ja sen integrointi pelikokemukseen. Google Play Game Service sekä Applen Game Center ovat myös sellaisia aiheita, mitkä ehdottomasti kaipaisivat lisäselvitystä. Ne ovat palveluita, mitkä voidaan liittää osaksi peliä. Käyttäjälle ne tarjoavat esimerkiksi tavan pelata samaa peliä usealla laitteella niin, että voidaan jatkaa peliä puhelimella joka on aloitettu tabletilla. Tämän lisäksi ne tarjoavat ennätystuloslistoja ja erilaisia pelaamisesta kertyviä saavutuksia, joita voi ystävien kanssa vertailla.

Osaan näistä kysymyksistä Cocos2d-x:n uusin versio tuo hieman helpotusta, mikä onkin erinomainen uutinen. Esimerkiksi vasta pidetyssä Cocos Developer -konferenssissa Google julkisti AdMob-palvelun tuen Cocos2d-x:lle. Näin kehittäjä voi liittää mainospalvelun osaksi peliään ja tätä kautta menestyä taloudellisesti (Cocos2d-x, 2014).

Mobiilialustojen osalta kamppailu näyttää olevan kuitenkin kahden kauppaa, Android- ja iOS -laitteet ovat oikeastaan ne ainoat varteenotettavat laitteet joille pelejä kannattaa kehittää. Windows Phone tulee niin kaukana jäljessä, ettei ainakaan sen ehdoilla kannata peliä kehittää. Voidaan ajatella, että Windows Phone -alustalla olisi vähiten kilpailua, mutta Windows Phone –käyttäjien määrä on niin pieni ja vielä vähemmän heitä, jotka olisivat valmiita maksamaan peleistä.

**LÄHTEET**

Chapman Cameron. 24.3.2010. A Short Guide to Open-Source and Similar Licenses. Luettu 20.10.2014.
http://www.smashingmagazine.com/2010/03/24/a-short-guide-to-open-source-and-similar-licenses/

Cocos2d-x. 2014. Audio formats supported by CocosDenshion on different platforms. Luettu 25.10.2014.
http://www.cocos2d-x.org/wiki/Audio_formats_supported_by_CocosDenshion_on_different_platforms

Cocos2d-x. About Us. Luettu 10.9.2014.
http://www.cocos2d-x.org/wiki/About_Us

Cocos2d-x. 2014. The Cocos Developer Conference (Fall) 2014 Succesfully Held – Google And Facebook Now On Board. Luettu 6.11.2014.
http://www.cocos2d-x.org/news/371

Cocos2d-x. 2014. Cocos2d-x v3.0 – What's new. Luettu 20.10.2014.
http://www.cocos2d-x.org/news/216

Engelbert, R. 2013. Cocos2d-x by Example Beginner's Guide. Birmingham, UK: Packt Publishing.

Marmalade. 2014. Marmalade Quick. Luettu 25.10.2014.
https://www.madewithmarmalade.com/products/quick

OpenSignal. 08/2014. Android Fragmentation Visualized. Luettu 10.10.2014.
http://opensignal.com/reports/2014/android-fragmentation/

Opensource.com. What is open source?. Luettu 10.10.2014.
http://opensource.com/resources/what-open-source

Pearson Dan. 22.10.2014. Mobile to become gaming's biggest market by 2015. Luettu 2.11.2014.
http://www.gamesindustry.biz/articles/2014-10-22-report-mobile-to-become-gamings-biggest-market-by-2015

Pitkänen Perttu. 14.9.2014. Onko Windows Phonen aallonpohja Suomessa jo ohi? Luettu 20.10.2014.
http://www.itviikko.fi/uutiset/2014/09/13/onko-windows-phonen-aallonpohja-suomessa-jo-ohi/201412654/7

Racketboy. 2012. Shmups 101: A Beginner's Guide to 2D Shooters. Luettu 1.10.2014.
http://www.racketboy.com/retro/shooters/shmups-101-a-beginners-guide-to-2d-shooters

Tomi T. Ahonen. 2014. Smartphone Top 10 for Q2 of 2014, same ole same ole... Luettu 20.11.2014.
http://communities-dominate.blogs.com/brands/2014/08/smartphone-top-10-for-q2-of-2014-same-ole-same-ole.html

Unity. 2014. Unity releases 2D tools with 4.3 update. Luettu 20.09.2014.
http://unity3d.com/company/public-relations/news/unity-releases-2d-tools-43-update

Wikipedia. Cocos2d. Luettu 20.10.2014.
http://en.wikipedia.org/wiki/Cocos2d

Wikipedia. 2014. Phablet. Luettu 20.10.2014.
http://en.wikipedia.org/wiki/Phablet