Ville Turunen

Indie-peliohjelmointi ja 2D Android -pelin rakentaminen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

13.2.2014

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Ville Turunen Indie-peliohjelmointi ja 2D Android -pelin rakentaminen 43 sivua + 2 liitettä 13.2.2014
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Juha Kämäri Lehtori Jorma Räty

Tämän opinnäytetyön aiheena on tarkastella indie-pelikehitystä ohjelmoijan näkökulmasta, Androidia kehitysalustana ja sitä, kuinka rakennetaan 2D-pulmanratkaisupeli.

Työ koostuu neljästä eri osasta. Näistä kaksi ensimmäistä osiota perustuu teoria-pohjaiseen tietoon, kolmas opinnäytetyön tekijän kokemusperäiseen toteutukseen ja neljäs ulkopuolisiin indie-pelinkehittäjien mielipiteisiin.

indie-peliohjelmointia käsittelevä osuus on rakennettu sujuvasti käsitellen tarpeellista tietoa ja sitä, mitä tämä tarkoittaa ja miten se toimii. Tässä osiossa myös perehdytään siihen, miten indie-pelin kehittäjä pystyy luomaa omaa uraa ja saamaan rahaa tällä alalla.

Android-sovelluskehitystä käsittelevä osuus taas kertoo Androidin kehitysvaiheista ohjelmointikielenä sekä mobiili- ja tablettialustana. Tämän lisäksi käydään läpi Android-sovelluksien jakelutapoja ja Androidin markkinaosuuksia.

Android-pelin toteutus pohjautuu opinnäytetyössä luotuun käytännön työhön, joka kehittyi oppaan tekemisestä selvästi monimutkaisempaan ratkaisuun. Toteutuksessa pyritään kertomaan tarkasti työn luontityylistä sekä varsinaisesta työstä. Työssä on käytetty ohjelmoinnin lisäksi matemaattisia laskuja, grafiikkaa ja äänituotantoa. Toteutuksessa on pyritty siihen, että lukija voi ymmärtää koodiesimerkit ilman ohjelmointitaustaa.

Viimeisessä osiossa on muiden indie-pelinkehittäjien kokemuksia ja mielipiteitä kiteytetty mielipidetiedustelun yhteenvetona. Mielipidetiedusteluun osallistui kymmenen ihmistä.

Avainsanat	Indie, Android, Peli

Author(s) Title Number of Pages Date	Ville Turunen Indie game programming and building 2D Android game 43 pages + 2 appendices 13 February 2014
Degree	Bachelor of Engineering
Degree Programme	Information Engineering
Specialisation option	Software Engineering
Instructor(s)	Juha Kämäri, Senior Lecturer Jorma Räty, Senior Lecturer

This thesis examines indie game development from programmer's point of view, using of Android development kit and the practice of building a 2D puzzle solving game.

The work consists of four parts. The first two sections cover theory-based knowledge. The third part discusses author's experience-based execution, and the fourth part examines the opinions of third-party game developers.

The section discussing indie game programming is smoothly comprised of handling necessary information about the subject, its meanings and the way it functions. This section also focuses on the practice indie game developers are able to create their own career and be successful in this business.

The part examining Android application development includes Android development phases as programming language for mobile and tablets. This part also covers distribution styles of Android application and Android's market shares.

Android game execution is based on the work which was done for this thesis. Developed game progressed from a tutorial to a much complex solution. The execution part is purposed to precisely represent the creation style of the work and the actual work itself. In addition to programming work, game development included mathematic calculations, graphics and audio production. The purpose of the implementation of this game was that the reader with no programming background could understand the code examples.

The last section was created from poll results. Experiences and opinions of other indie game developers were summarized. Ten people took part to the poll.

Keywords	Indie, Android, Game

Sisällys

Lyhenteet

2D	2 Dimensions. Digitaalinen kuva tai teksti, joka sisältää kaksi ulottuvuutta.
ADT	Android Development Tools. Eclipse-ohjelman lisäohjelma, jonka avulla on tarkoitus luoda Android sovelluksia.
Apache ANT	Apache Another Neat Tool. Ohjelmistotyökalu, joka mahdollistaa Java-ohjelmien rakentamisen komentorivin kautta.
HTML	HyperText Markup Language. Maailman laajuisesti käytetyin merkintäkieli, jota käytetään Internet sivujen selaamiseen.
iOS	iPhone Operating System. Applen käyttöjärjestelmä, jota käytetään iPhone kännyköissä, sekä iPad tableteissa.
Java EE	Java Platform, Enterprise Edition. Oraclen kehittämä Java ohjelmointikieli, joka on kehitetty yritysympäristölle.
JIRA	Gojira. Atlassianin tekemä tehtävienhallintaohjelmisto.
JPG/JPEG	Joint Photographic Experts Group. Kuvien tallennusformaatti.
LAN	Local Area Network. Lähiverkko, jonka sisällä on useampi kone samassa sisäverkossa.
NDK	Native Development Kit. Kehitystyökalu, joka käyttää hyväksi C-ohjelmointikieltä.
OS X	Operating System X. Applen tietokoneiden käyttöjärjestelmä.
PNG	Portable Network Graphics. Kuvien tallennusformaatti.
PS	PlayStation. Sonyn luoma videopelikonsolimerkki.
SDK	Software Development Kit. Kehitystyökalu, jonka avulla pystytään kehittää sovelluksia. Androidissa käytetään hyväksi Java-pohjaista SDK:ta.

Tekes	Innovaatiorahoituskeskus Tekes. Suomen valtion virasto, joka aktivoi ja rahoittaa tutkimus- ja kehitysprojekteja.
XML	Extensible Markup Language. HTML:ää muistuttava merkintäkieli, jonka ideana on muotoilla ohjelmakieli koneelle ja ihmiselle luettavaan muotoon.

1 Johdanto

Suomessa peliala on kasvanut huimasti viime vuosina, mutta harva tietää sen olevan jo 30 vuotta vanha. Tekniikka on muuttunut hurjasti, ja muun muassa mobiili- ja tablettikehitys on tuonut indie-kulttuuria pinnalle entistä enemmän.

Tämä opinnäytetyö koskee indie-pelien kehitysvaiheita ja sellaisen ohjelmointia. Työssä käytetään hyväksi Android-alustaa, joka on maailman suosituin mobiilikäyttöjärjestelmä. Työssä perehdytään kolmeen osa-alueeseen: indie-peliohjelmointiin, Android-käyttöjärjestelmään ja 2D-pelin toteutukseen. Lopuksi vielä työn ohessa on mielipidetiedustelu muilta indie-pelintekijöiltä.

Indie-käsitteen käyttöä on opinnäytetyössä rajattu jonkin verran. Vaikka sillä usein kuvataan muun muassa tietyntyyppistä pelikehitystä koko toimialana, keskitytään tässä työssä siihen vain peliohjelmoinnin näkökulmasta. Toisaalta selvyyden vuoksi tähän sisältyy, mitä indie tarkoittaa ja kuinka indie toimii, mutta kaikkea indie-konseptin sisältöä ei käydä läpi.

Androidiin liittyvät käsitteet keskittyvät Android-käyttöjärjestelmään, mutta työssä käsitellään myös Androidin kehitystä mobiili- ja tablettikäyttöjärjestelmänä. Työssä perustellaan, miksi haluan kehittää peliä ensisijaisesti Android-alustalle, mitä hyötyjä Android alustana antaa ja mitä tulee ottaa huomioon kehittäessä Android-alustalle.

Peli on toteutettu ADT-paketin avulla, joka toimii Eclipsen käyttöliittymällä. Pelissä käytetään Java-ohjelmointikieltä sekä XML-muotoilua. Pelin toteutuksessa käydään perusteellisesti läpi, kuinka peli tehtiin ja kuinka pelin mekaniikka sekä käyttöliittymä on rakennettu ohjelmallisesti. Lisäksi työssä käydään läpi pelin testaamisen vaiheita ja sitä, miksi testaaminen on tärkeää. Toteutuksen lopussa kerrotaan tulevaisuuden suunnitelmista kuten markkinoinnista, promotoinnista, jatkosuunnitelmista ja ylläpidosta.

Viimeisin opinnäytetyön osio koostuu muiden indie-pelintekijöiden mielipidetiedustelusta ja näiden vastauksien läpikäymisestä ja referoinnista.

2 Indie-peliohjelmointi

Indie-sanan virallinen määritelmä tarkoittaa itsenäistä. Tämä sana on tullut erittäin suosituksi pelinkehityksen saralla, joten yleensä indie käsitetään kansainvälisesti synonyymiksi ”indie game development” -termille. Indie-peliohjelmointi on osa tätä indie-käsitettä, mutta rajaa käsitteen koskemaan nimenomaan pelin kehitystä ja mahdollisuuksia ohjelmoijan näkökulmasta. Indie ei sanana ole muutenkaan täysin selvä tai yksiselitteinen. Alkuperäinen sanan määritelmä lähtee kannalta, että jokin teos tai tuote on itsenäisesti tehty riippumattomana ulkopuolisista tahoista. Nykyään indie-sana kuvaa lähinnä riippumattomuutta ja itsenäistä hallintaa. [1.]

Indie-pelin perustana on, että se on tehty yksin tai pienessä porukassa ilman julkaisijaa. Suurin hyöty pelin julkaisussa ilman julkaisijaa on se, että kuka tahansa voi aloittaa oman yrityksen tai toiminimen käyttäen hyväksi omaa budjettia ja aikaa oman osaamisen puitteissa. [1.]

2.1 Indie-pelin toteutustavat

Pelin tekemisen ei välttämättä tarvitse olla pitkäaikainen työ. Toteutustapoja on keksitty monia erilaisia, jotka jaetaan usein projekteiksi tai prosesseiksi. Indie-pelin rakentaminen on yleensä nopea prosessi, mutta pelimekaniikan luonti ei vielä tee pelistä valmista. Normaalisti peli vaatii graafista näkymää, juonen, testausta ja lisäominaisuuksia, jotta peli kokemuksena olisi halutulla tasolla. Ohjelmointivirheet voivat pilata pelin jo julkaisussa, mikä aiheuttaa negatiivista palautetta, joka taas vaikuttaa pelin markkinointiin. Tämän takia yleensä tekijät haluavat hioa peliään moneen kertaa läpi, jotta pelaaja saa pelistä halutunlaisen kokemuksen.

Nopeasti rakennettuja pelejä tehdään myös harrastuksena. Yksi tällainen tapa on tehdä peli jameina, eli kansainvälisesti ”game jam”. Näitä tapahtumia pidetään satoja vuodessa ja niihin voi osallistua joko verkon välityksellä tai paikanpäällä. Peli-jamien ideana on toteuttaa peli yleensä 24–48 tunnin sisällä valmiiksi. Pelin toteutus on tällöin yleensä hyvin yksinkertainen, eikä se vastaa kaupallisia pelejä. Peli-jameissa on myös mahdollista tavata muita pelinkehityksestä kiinnostuneita henkilöitä. [2.]

2.1.1 Projekti- ja prosessityylit

Projekti koostuu useammasta prosessityövaiheesta, jotka yleensä jaetaan seuraavasti: määrittely, suunnittelu, toteutus, testaus, julkaisu ja ylläpito. Projektityyppejä käytetään melkein kaikissa tietotekniikan yrityksissä, joissa kehitetään uutta tuotantoa. Projektimalli takaa työn seurantaan varman pohjustan, budjetin varojen hallinnan ja ajan seurannan.

Indie-peliohjelmoinnissa käytettävä projektimalli yleensä eroaa yritysten projektimallista. Tämä johtuu siitä, että tekijöitä on vähäinen määrä. Tästä huolimatta projektimallia käytetään määrittelyn laatimiseen. Yleisiä määrityksiä ovat pelin tekemiseen takaraja, lupaus pelaajille kertoa pelin etenemisestä ja selvyys kokonaistyöstä. Indie-peliprojekteissa on yleistä, että määrittelyt eivät pidä lopulta paikkansa, minkä takia niitä suositellaan tuplaamaan tai moninkertaistamaan. [3.]

Mikäli pelin tekijä päättää tehdä työnsä pelkästään prosessinomaisella työtavalla, tarkoittaa se sitä, että peli valmistuu sitten kun valmistuu. Prosessityöskentelytapa suosii pelinkehitystä pelaajien näkökulmasta. Tekijät voivat antaa pelaajien pelata peliä jo varhaisessa vaiheessa, minkä avulla pelaajat taas vuorostaan voivat auttaa tekijää pelin kehityksessä. Vuoroittaisessa kommunikoinnissa pelaajat voivat kertoa mistä he pitivät pelissä sekä mitä pitäisi muuttaa. Prosessityyli sopii hyvin, jos tekijöitä on yksi tai kaksi, sillä he voivat hyvin joustavasti rakentaa peliä riippuen siitä, missä on suurin tarve kehitykselle. [3.]

2.1.2 Pelin alusta

Pelin alustan valitseminen voi olla sekä ongelma että mahdollisuus. Ongelma voi tulla esille siinä, ettei tietylle alustalle kehitetty peli tuota yhtä hyvin, kuin jos se tehtäisiin toiselle alustalle. Mahdollisuutena taas tekijä voi päästä selvästi vähemmällä työllä, mikä helpottaa päätöstä siitä, kannattaako pelin kehitystä jatkaa muille alustoille. Nykyään indie-pelit on suurimmaksi osaksi rakennettu mobiili- ja tablettialustoille, koska nämä ovat halpoja tuottaa sekä voidaan tehdä lyhyessä ajassa. Tästä huolimatta indie-pelejä tehdään melkein kaikille käyttöjärjestelmille kuten Androidille, iOS:lle, Windows Phonelle, Linuxille, OS X:lle, Windows-tuoteperheelle, Playstation 3:lle, 4:lle ja Vitalle, Xbox 360:lle, Xbox Onelle ja niin edelleen. Indie-peli on myös mahdollista luoda alustariippumattomaksi. Alustasta riippumattomat pelit vaativat yhteisen ohjelmointirajapin-

nan, joka toimii kaikilla alustoilla samalla tavalla. Tällaisia ovat esimerkiksi Adobe Flash ja HTML5.

Nykyään pelimoottorin luonti ei ole enää pakollista, sillä pelinkehittäjille on luotu ohjelmia, joiden avulla voi luoda pelejä ilman perinteistä ohjelmointia. Näitä ovat esimerkiksi Unity3D, Game Maker, Construct 2 ja Stencyl. Tämän kaltaiset ohjelmat helpottavat pelin rakentamista todella paljon sekä mahdollistavat pelin luomisen usealle alustalle. Tällöin tosin pelin tekeminen aiheuttaa usein rajoitteita tai maksuja. [4.]

2.1.3 Toteutuserot julkaisijan kautta toteutettaviin

Ennen indie-pelikehityksen nousukautta melkein kaikki tunnetut pelit tehtiin julkaisijan kanssa. Prosessi toimi useammassa osassa, joista ensimmäisenä luotiin pelistä prototyyppi tai demo. Tämä vietiin julkaisijalle esiteltäväksi tarjouksen kera, minkä jälkeen julkaisija päätti, haluaako ostaa peliyrityksen tekemään pelin valmiiksi asti. Mikäli julkaisija oli kiinnostunut pelistä, käytiin läpi määrittelyvaihe, missä määriteltiin tärkeät seikat liittyen pelin sisältöön, aikatauluun ja budjettiin. Kummankin hyväksyttyä sopimuksen peliyritys oli sitoutunut tekemään pelin valmiiksi julkaisijan määrittelemää sopimusta noudattaen. Pelin valmistuttua kävi usein niin, että julkaisija keräsi voiton, mikäli peli ei menestynyt odotetulla tavalla. Toisaalta mikäli sen menestys oli kiitettävää, saattoi peliyritys saada bonukset lisäansiona. [5.]

Suurimmat erot perinteisen pelinkehityksen ja indie-pelinkehityksen välillä ovat budjetti, aika ja se, kuka omistaa pelin. Perinteisessä mallissa mikäli pelin tekeminen ei ole julkaisijan vaatiman tason mukainen, on julkaisijalla mahdollisuus etsiä toinen peliyritys suorittamaan pelin tekeminen loppuun asti. [5.]

2.2 Ansaintamallit

Indie-peliohjelmoinnin yksi suurimmista hyödyistä löytyy rahanteossa. Tässä on sekä hyviä että huonoja puolia verrattuna julkaisijan kautta saatuun tulonlähteeseen. Indie-peleissä raha tulee joko suoraan käyttäjiltä, valtion avustuksena, yrityksiltä, sijoittajilta tai lainapääomana.

2.2.1 Itsenäiset ansaintamallit

Pelien itsenäisiä ansaintamalleja ovat pelin myynti, mikromaksut, mainokset, ennakkomyynti ja joukkorahoitus. Suurin osa näistä liittyy pelin myymiseen, mutta myyntitavat eroavat toisistaan. Ennen digitaalista jälleenmyyntiä oli indie-pelien myynti erittäin vaikeaa. Jälleenmyynti tapahtui kahdella tavalla: postimyynnillä tai fyysisesti kaupan hyllyllä. Kaupat kuitenkaan eivät usein ottaneet pelejä suoraan myyntiin, vaan he yleensä vaativat, että peli myydään ensin jälleenmyyjille tai maahantuojille. Nykyään asiat ovat muuttuneet ja lähestulkoon kaikki indie-pelit myydään digitaalisesti.

Myynti digitaalisesti on erittäin yksinkertaista, mutta se ei takaa suurta rahallista voittoa. Pelin pitää erottua joukosta, jotta sitä halutaan pelata. Aloitteleville indie-pelintekijöille suositellaan käytettäväksi myynnin lisäksi myös toista ansaintamallia, mainoksia. Ilmainen peli innostaa pelaajia koettamaan peliä, jolloin pelin jakelu edistyy sekä pelin omistaja saa mainoksista rahaa. Näin ollen kehittäjä ansaitsee rahaa mainoksien jakajilta eikä pelaajilta. [6.]

Nykyään kaikki pelit eivät tule yhdessä paketissa, mikä taas mahdollistaa uudenlaisen myyntitavan: mikromaksun. Mikromaksut koostuvat hyvin pienestä ominaisuudesta tai palvelusta pientä rahallista korvausta vastaan. Nämä ovat yleensä uusia kenttiä, uusia vaatteita pelissä tai pelin sisäistä rahaa. Mikromaksujen on väitetty olevan petollinen tapa ansaita rahaa, koska ne saattavat aiheuttaa peliriippuvuutta samalla tavalla kuin peliautomaatit. [7.]

Ennakkomyyntiä voidaan tehdä kahdella eri tavalla. Ensimmäinen tapa on myydä peliä ennakkoon, jonka jälkeen tekijä on velvollinen luomaan pelin valmiiksi tiettyyn aikarajaan mennessä. Pelin pitää myös sisältää luvatut ominaisuudet. Toinen ennakkomyyntitapa on joukkorahoitus. Tämä on ennakkoon pyydetty rahamäärä peliä, oheistuotteita, palvelua, tapaamista tai ominaisuutta vastaan. Joukkorahoitus on saanut mediassa paljon huomiota, mutta Suomessa ilmiö on vielä uusi. Negatiivinen puoli joukkorahoituksessa on, ettei siihen ole tehty lakia, joka täsmentäisi tai rajaisi sen epäselviä kohtia kuten esimerkiksi pelintekijän velvoitteita. [8.]

Joukkorahoitus toimii niin, että pelintekijä pyytää halutun summan rahaa, mikä käyttäjien pitää kollektiivisesti ylittää. Mikäli peliä ei osteta ennakkoon tarpeeksi paljon eikä maalia saada täyteen, tulee joukkorahoitussivuston maksaa kaikki rahat takaisin käyt-

täjille. Yleisimmät joukkorahoituksen sivustot ovat kickstarter.com sekä indiegogo.com. [8.]

2.2.2 Ulkopuoliset ansaintamallit

Suomessa valtio tukee yrityksien perustajia, ja erityisesti indie-peliohjelmoijille sopii hyvin Kansaneläkelaitoksen starttiraha. Se ei ole paljoa kuukaudessa, mutta indie-pelin tekeminen onnistuu myös vähäisellä budjetilla. Valtio ei pyydä tukia jälkeenpäin takaisin, minkä takia se on erittäin hyvä valinta pienellä budjetilla rakennetun indie-pelin kehitykseen. [9.]

Toinen Suomen valtion tukema rahoitus pelialalle on Tekes:in järjestämä skene-ohjelma, joka voi parhaimmillaan tukea yritysten kehitystä yhdellä miljoonalla eurolla. Tämä kuitenkin vaatii sen, että Tekes myöntää yritykselle vain osarahoituksen, eikä rahoita projektia täysin itse. Tekes myöntää myös lainaratkaisuja. Esimerkiksi Suomen toiseksi suurin peliyritys, Supercell, otti lainan Tekes:ltä yrityksen alkuvaiheessa. [10.]

Laina kuitenkaan ei ole täysin riskitön vaihtoehto, minkä takia yritykset usein myyvät osan tuotteistaan tai yrityksestään sijoittajille. Tämä tuo paljon pienemmän riskin kuin lainan ottaminen. Se on tosin usein hankalaa, sillä harvemmin sijoittajat kiinnostuvat pelistä ennen kuin se on julkaisuvalmis. Toki auttaa paljon, jos tekijät ovat jo ennestään tunnettuja indie-puolella.

2.3 Indie-peliohjelmoinnin hyödyt ja haitat

Vaikka indie-pelin tekeminen kuulostaa houkuttelevalta monella tavalla, niin yleensä jää tietämättä tämän monesta vaaroista. Tehdyistä indie-peleistä vain pieni prosentti saa tuottoa omasta pelistänsä, joten hyödyt eivät ole välttämättä yhtä suuret kuin ajankäyttö tämän suhteen. Mikäli haluaa tehdä oman indie-pelin, olisi syytä selvittää tämän käsitteen hyödyt ja haitat.

Hyödyt:

- Aikataulu on tekijän mukainen.

- Pienemmät seuraamukset, mikäli epäonnistuu ja mahdollisuus luoda omaa visiota.

- Pärjää pienemmällä rahoituksella tai jopa ilman rahoitusta.

- Ei ole sidottu julkaisijan antamiin rajoituksiin.

- Raha tulee tekijöille, eikä julkaisijoille.

Haitat:

- Aloittelijat eivät pärjää kovinkaan hyvin ja julkaisu saattaa epäonnistua.

- Suunnittelu ja toteutus eivät ole yleensä yhtä hyvää tasoa kuin julkaisijan kautta tehdyissä peleissä.

- Markkinointi vaikeaa, jos ei ole tuottoja.

- Rahoituksen saaminen on hankalaa, mikäli sille on tarvetta.

3 Android-sovelluskehitys

Androidin sovelluskehityksen täysi tuki on rakennettu Eclipselle ja ADT-lisäohjelmalle, joka toimii Javalla ja XML:llä. Näiden ohjelmien lisäksi tarvitaan myös Android SDK -paketti, joka sisältää ohjelman kehitystyökalut. Kuitenkaan tämä ei ole ainut tapa, millä Android-sovelluksia voi tehdä. Androidin sovelluskehittäminen onnistuu myös monella muulla ohjelmalla. Esimerkiksi valmiilla moottoreilla on mahdollisuus kehittää Android-ohjelmia. On myös muita ohjelmointikielimahdollisuuksia, kuten Android SDK:n sijaan voi käyttää Android NDK:ta, joka taas käyttää hyväkseen C/C++-ohjelmointikieltä. Kaiken lisäksi on myös mahdollista käyttää eri käyttöliittymää Eclipsen sijaan, kuten esimerkiksi Visual Studio:ta VisualGDB-työkalun avulla tai komentoriviä käyttäen hyväksi Apache ANT -työkalua. [11; 12; 13.]

Uusin tapa kehittää Android-ohjelmia on Android Studiolla, joka toimii eri työympäristössä Eclipsen sijaan. Se on Googlen oma ohjelma, joka luultavasti tulee tulevaisuu-

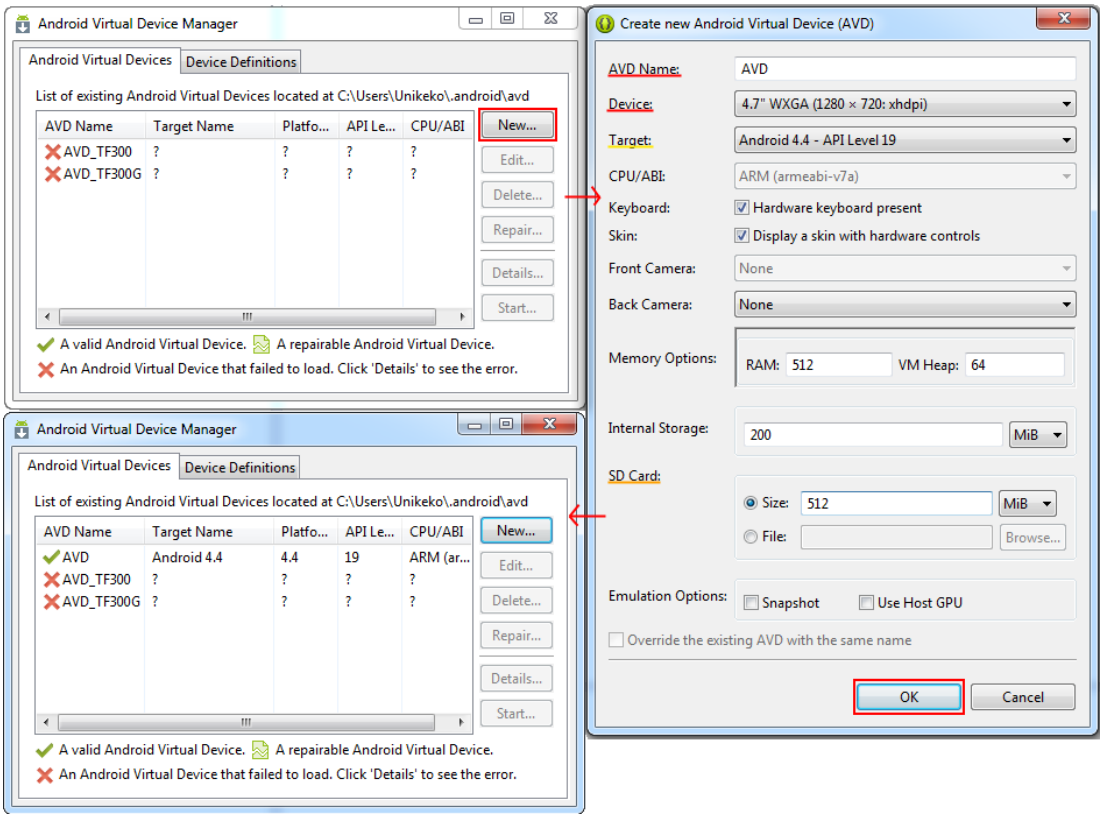
dessa syrjäyttämään Eclipsen, koska sen käyttöliittymä sopii paremmin usean näytön testaamiseen, sekä se on selvempi kuin Eclipse. Tällä hetkellä Android Studio on vasta aikaisessa ennakkovaiheessa, joten kaikki ominaisuudet eivät vielä toimi tai toimivat vajavaisesti. [14.]

3.1 ADT-paketti

Android on kehittänyt ADT-paketin, joka sisältää useamman tarvittavan työkalun, joita tarvitaan Android-ohjelmointiin. [15.]

- Eclipse + ADT-lisäohjelman
- Android SDK -paketin
- Android-sovellusalusta-työkalut
- viimeisimmän Android-sovellusalustan
- viimeisimmän Android järjestelmä emulaattorin.

Kun ADT-paketti on ladattu koneelle sekä asennettu haluttuun paikkaan, tarvitsee enää luoda emulaattori käyttöön Android Virtual Device Managerin kautta. Tämän jälkeen Android-ympäristö on valmis käytettäväksi (kuva 1).



Kuva 1. Emulaattorin luonti Android Virtual Device Managerin avulla.

3.1.1 Android-projekti

Android-sovellus koostuu tietystä rakennetyypistä, projektista. Tämä sisältää jokaiselle osiolle omat osionsa ja osaa lukea ainoastaan, jos projekti on rakennettu oikealla tavalla. Tärkeimmät kansiot ja tiedostot kehityksen kannalta ovat: src/-, bin/-, res/-, libs/ -kansiot sekä AndroidManifest.xml-tiedosto. [16.]

src/-kansio sisältää Java-koodit ja varsinaisen ohjelman. Toteutettavat Java-tiedostot luodaan Activityinä, mutta tämän lisäksi src/ -kansiossa voi olla myös muita lähdekoodeja, esim. olioita. [16.]

bin/-kansio sisältää Android-sovelluksen toteutuksen, mistä tekijälle tarpeellinen on ainoastaan ”.apk”-tiedosto. Tämä sisältää sovelluksen paketissa, jonka Android osaa purkaa omaan käyttöönsä. [16.]

res/-kansio sisältää useamman kansion, joista jokainen liittyy ohjelman sommitteluun ja siihen, mitä ohjelma voi käyttää resursseina. Näistä tärkein on /layout-kansio, joka sisältää XML-tiedostot, joilla on mahdollista sommitella ulkoasu ohjelmalle, mikäli haluaa käyttää XML-muotoilua. [16.]

libs/-kansio sisältää yksityisrajapinnat, joiden avulla on mahdollista käyttää ulkopuolisia metodeja, joita voi lisätä projektin ominaisuuksista ”.jar”-tiedostoina. [16.]

AndroidManifest.xml kontrolloi ohjelman toimivuutta ja lupia, joita ohjelma vaatii sen käyttöön, jotta tämä voidaan asentaa Androidille. Tiedosto sisältää myös tiedon siitä, että millä versioilla ohjelmaa voidaan käyttää. Esimerkiksi Android 2.2 pystyy käyttämään ainoastaan ominaisuuksia, jotka ovat ohjelmointirajapinnaltaan kerrostasolla 1-8. [16.]

3.1.2 Java

Java on laitteistosta riippumaton oliopohjainen ohjelmointikieli, jota käytetään ohjelmien luonnissa tai ohjelmistoalustoissa. Javan kehitti Sun Microsystem 1990-luvun alussa ja sitä käytettiin alun perin vuonna 1995 WWW-sivuille luotavien applettien tekemiseen, mikä teki Javasta suurmenestyksen. Javan käyttö nykyään on laajentunut huimasti 1990-luvusta. Nykyään Java-alusta on käytössä n. 3,8 miljardissa laitteessa mobiililaitteista supertietokoneisiin. Nykyään Javan omistaa Oracle Corporation, joka fuusioitui Sun Microsystemin kanssa 2010 vuoden alussa. [17.]

Androidissa Java toimii eri tavalla kuin Oraclen Java. Google päätti käyttää Javaa avainasemassa Android-käyttöjärjestelmässään, mutta he rakensivat oman Android-ohjelmointirajapinnan, käyttäen hyväksi Javan ohjelmointirajapintaa. Näin ollen nämä ovat rakenteiltaan samanlaisia, mutta täysin kaksi eri Java-ohjelmointirajapintaa. Tästä syntyi vuonna 2012 lakisyyte Googlea vastaan, mutta tämä hylättiin perusteella: ”Ohjelmointirajapintaa ei voida suojata tekijäoikeudella.” [17.]

3.1.3 XML

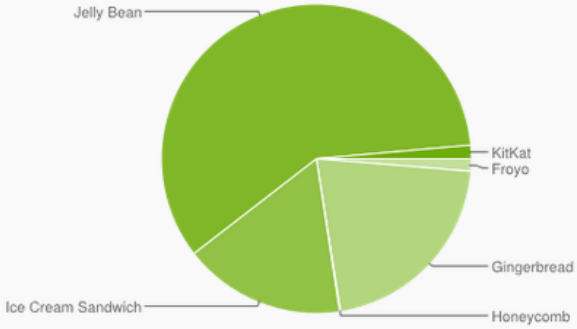
Extensible Markup Language, eli XML on HTML:ää muistuttava merkintäkieli, jonka ideana on muotoilla koodi koneelle ja ihmiselle luettavaan muotoon. XML on kehitetty vuonna 1996, ja sen standardin on tuottanut WC3. XML:stä on kaksi eri standardia: 1.0 ja 1.1. Näistä yleensä käytetään 1.0-versiota, joka on viides versio ja päivitetty vuonna 2008. XML:llä ei ole varsinaista omistajaa, koska se on luotu avoimessa formaatissa. [18.]

Androidissa XML:ää käytetään muotoilussa. XML ei ole pakollinen ominaisuus, koska Androidissa voidaan käyttää omaa piirto-ominaisuutta View-luokan avulla. Kuitenkin XML on luotu sitä varten Androidiin, että muotoilu olisi selvästi helpompaa. Tällöin myös pystytään käyttämään valmiita ominaisuuksia hyväksi vähemmällä työllä.

3.1.4 Android-ohjelmointirajapinta

Nykyään ei voida määritellä Android-ohjelmia täysin samanlaisiksi, koska uudistuksien myötä Android-ohjelmointirajapinta on mahdollistanut uusia ominaisuuksia. Jokainen ohjelmointirajapinta määritellään kerroslukuna, jotka ovat standardina Android-versioissa. Esimerkiksi android.animation -luokka on luotu ohjelmointirajapintakerroslukuun 11, joten tätä voi käyttää vain Androidin versiossa 3.0 tai uudemmassa. Tämän takia suositellaan olemaan tarkkana sen kanssa, mitä ominaisuuksia haluaa käyttää, koska kaikki mobiililaitteet eivät kykene päivittymään versioon 3.0 tai uudempaan (kuva 2). [19.]

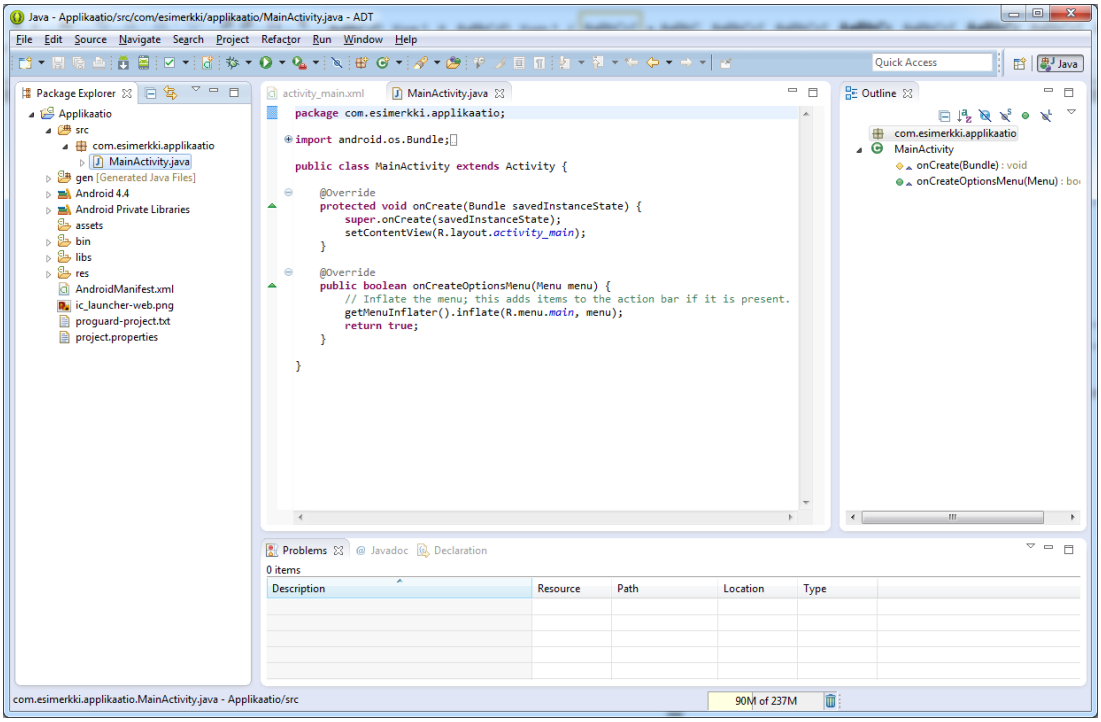
Versio	Version nimi	API	Jakauma
2.2	Froyo	8	1.3%
2.3.3 - 2.3.7	Gingerbread	10	21.2%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	16.9%
4.1.x	Jelly Bean	16	35.9%
4.2.x		17	15.4%
4.3		18	7.8%
4.4	KitKat	19	1.4%



Kuva 2. Android-versioiden vertailu. (tehty: 8.1.2014) [20.]

3.2 ADT-paketin käyttöliittymä

ADT-paketissa tuleva käyttöliittymä on hieman riisuttu versio normaalista Eclipsestä (kuva 3). Tämä on tarkoitettu kevyeksi Eclipse-versioksi, joka ei sisällä Java EE -ominaisuuksia tai Eclipsen omaa kauppaa, josta voi ladata lisäohjelmia. Puutteista huolimatta tämä on toimiva paketti ja sisältää kaiken tarpeellisen Android-ohjelman luomiseen.

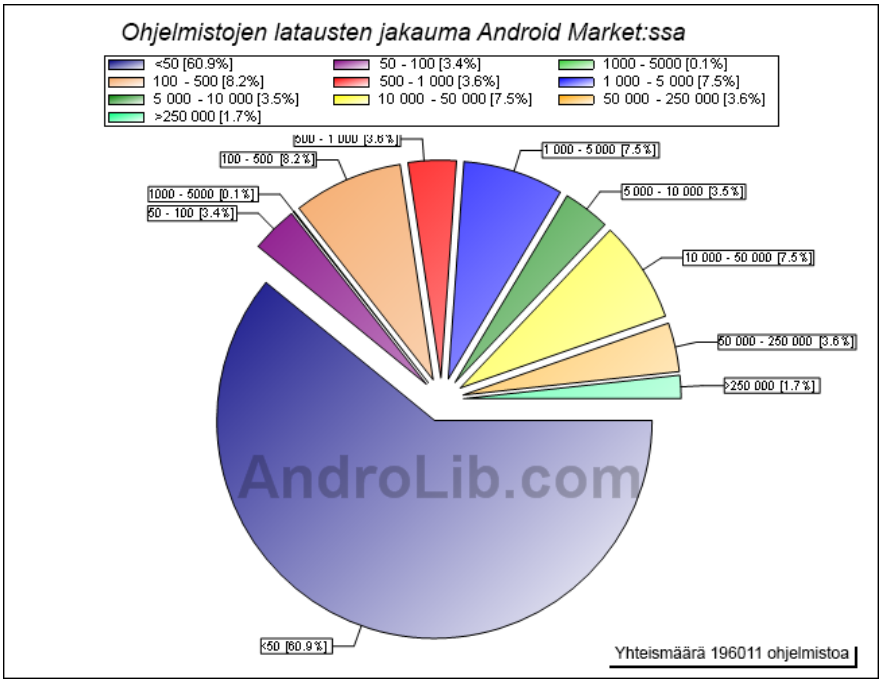


Kuva 3. Kuvakaappaus ADT-paketin käyttöliittymästä: Eclipse.

3.3 Play-kauppa ja muu levitys

Play-kauppa on Googlen oma digitaalinen jakelukanava, mistä voi ladata sekä ostaa sovelluksia, kirjoja, musiikkia, laitteita, lehtiä, elokuvia ja tv-ohjelmia. Suomessa tämä on rajoitettu pelkästään sovelluksiin, kirjoihin ja musiikkiin. Play-kauppa oli nimeltään Android Market, mikä on vielä toiminnassa vanhemmissa Android-versioissa kuin 2.2. Play-kaupassa julkaisu vaatii Google-tilin, tekijäsopimuksen hyväksymisen ja lisäksi rekisteröinti maksaa 25 dollaria. Tämän jälkeen on mahdollista luoda ohjelmia Play-kauppaan julkaisemattomana tai julkaistuina. Play-kaupassa sovelluksen myynnistä tekijä saa 70 % hinnan osuudesta. [21; 22.]

Play-kaupassa kaikkien sovellusten arvioitu kokonaislatausmäärä on yli 50 miljardia, mutta tämä ei tarkoita, että olisi helppo saada paljon latauksia. AndroLib-sivuston mukaan keskimäärin kolme viidesosaa Play-kaupan sovelluksista on saanut vähintään 50 latausta, joista noin kaksi kolmasosaa koostuu ilmaisista latauksista. AndroLib-sivuston vertaukset koskevat alle viidesosa Play-kaupan sovelluksista, mutta tästä tutkimuksesta saadaan hyvin näkemystä, kuinka ohjelmat pärjäävät Play-kaupassa. Toinen AndroLib-sivuston vertailu koskee Play-kaupan sovelluksia, missä sovelluksista ainoastaan vaan 15,5 % on pelejä (kuva 4). [21; 23; 24; 25.]



Kuva 4. Play-Kaupan ohjelmien latausmäärät [23.]

Play-kaupan lisäksi on myös muita digitaalisia jakelukanavia. Samsung on tehnyt oman ”Samsung Apps” -jakelukanavan, joka on asennettu ainoastaan Samsungin puhelimille. Samsung ei kuitenkaan tarjoa samalla tavalla musiikkia tai kirjoja kuin Play-kauppa, mutta sähköisiä kirjoja on mahdollista ladata sovelluksen muodossa. Samsung Apps ei veloita tekijämaksuja vaan sallii ilmaisen jakelun. Sovelluksen myynnistä saa saman verran kuin Play-kaupassa, eli 70 %. [26.]

Kolmannen osapuolen jakelukanavia on luotu useita, joista suosituin sisältää yli 700 000 sovellusta, mutta ehkä tunnetuin näistä on Amazon Appstore. Amazon ei suinkaan ole vielä tunnettu Android-sovelluksista, vaan tuli tunnetuksi alun perin kirjakustantamona ja verkkokirjakauppana. Tämän jälkeen se on laajentanut valikoimaa moninkertaisesti DVD-levyistä elintarviketuotteisiin. Nykyään Amazon harjoittaa myös digitaalista Android sovellusten jakelua. Jakelu on ilmaista ensimmäisen vuoden, jonka jälkeen tämä maksaa 99 dollaria vuodelta. Amazon Appstoressa tekijä saa saman osuuden kuin Play-kaupassa, eli 70 %. [26.]

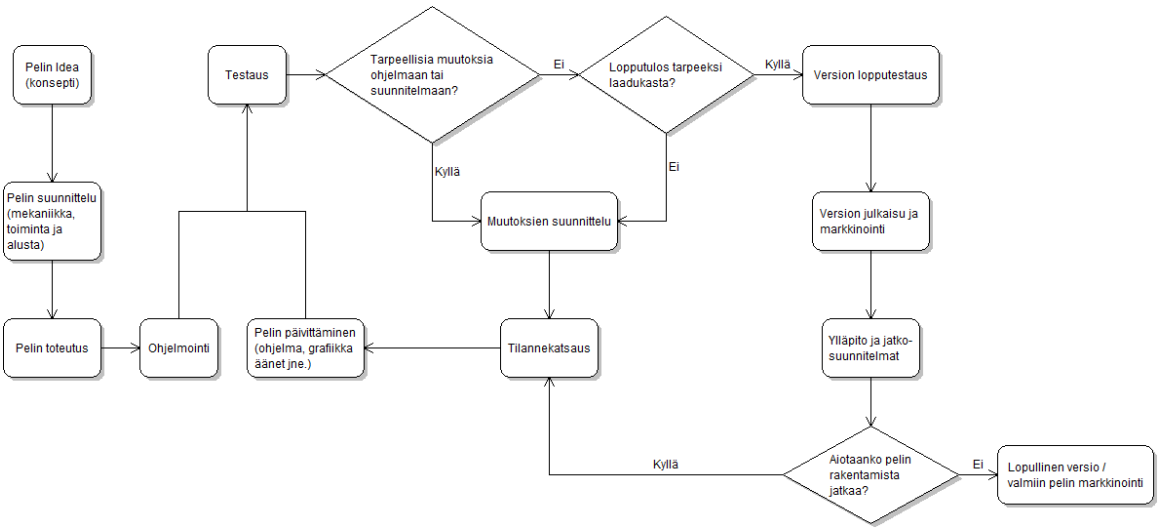
4 Android-pelin toteutus

Tämän opinnäytetyön idea syntyi halusta tehdä peli sekä samalla käsitellä indie-tyylistä pelintekoa, erityisesti keskittyen Androidiin. Työn idea syntyi vuoden 2012 syksyllä sil-

loisessa työssä koulutusaikoina taukoa viettäessä. Tämän jälkeen alkoi hahmottua suunnitelma siitä, miten peli tulisi toimimaan yksinkertaisilla metodeilla, mutta monimutkaisilla ratkaisuilla. Inspiroituneena ”Mr. Block”- ja ”Move the Box” -peleistä tarkoituksena oli kehittää peli, joka käyttäisi hyväksi uusia ratkaisutapoja sekä painovoimaa eri tavalla kuin muut pelit käyttävät. Peliä ei uskalla kutsua ainutlaatuiseksi, sillä on hyvin mahdollista, että samaa pelityyppiä on käytetty jo aiemmin. Tästä huolimatta itse en ole vielä törmännyt samanlaiseen Android-alustalla.

4.1 Pelin toteutustapa

Pelin toteutustapa muistuttaa hybridi-mallia projekti- ja prosessi tavoiltaan (kuva 5). Tämä johtuu siitä, että työtä pyrittiin tekemään projektina, pysyen määrätyissä aikatauluissa ja tehden tietyt asiat kerrallaan. Tästä huolimatta vaihteleva tekemisen mielihalu ajoi peliä eteenpäin sekä nopealla että hitaalla tavalla niin, että se muistutti enemmän prosessinomaista työskentelyä. Suunnittelu oli melko vähäistä ennen pelin rakentamista. Syy tähän oli se, että aikeissa oli kehittää peliä ketterästi. Alussa konsepti oli melko perusteellisesti hahmoteltu sekä hiottu suunnitelmatasolla, mutta juoni oli vielä keskeneräinen. Testaus toimi jatkuvassa integroinnissa teon yhteydessä, ja ohjelmointivirheet korjattiin lähes saman tien niiden ilmaantuessa. Tällä hetkellä pelissä on vielä muutama ohjelmallinen virhe, mutta syy johtuu siitä, ettei kaikkia ominaisuuksia ole otettu käyttöön.



Kuva 5. Pelin toteutuksen prosessikaavio

4.2 Suunnittelu

Normaalissa pelinkehityksessä toteutetaan alussa suunnitteluvaihe. Tämä kestää yleensä kauemmin kuin olisi tarpeen, koska pyritään jo alusta lähtien käymään läpi kaikki oleellinen pelin rakentamiseen liittyen. Ketterissä projektimalleissa pyritään hieman supistamaan suunnittelua, sillä se kohdistuu haluttuun päämäärään eikä välttämättä niinkään itse toteutukseen. Tämän pelin kehityksessä käytettiin hyvin vähäistä alkusuunnittelua. Suunnittelu oli itse asiassa hyvin samanlaista, kuin mitä käytetään aikaisemmin läpikäydyssä "Game Jam" -tyylisessä tapahtumassa. Tällöin suunnittelu pysyy nopeana ja suunnitelmat vaihtuvat kehityksen aikana.

4.2.1 Konsepti ja juoni

Pelin konsepti on yksinkertainen pulmanratkaisupeli, mutta ilman minkäänlaisia neuvoja pelissä, mikä voi tuottaa hankaluuksia päästä kenttiä läpi. Ideana on päästä alkupisteestä loppupisteeseen käyttäen hyväksi liikkumista sekä painovoimaa. Painovoima toimii ainoastaan, jos on alueella, jossa tämä on käytössä. Muuten pelissä on painovoimaton tila, jossa voi liikkua ihan mihin suuntaan tahansa, kunhan ei yritä seinän läpi mennä. Seinät ympäröivät kenttiä ja rajoittavat liikkumista. Seinät myös rajoittavat painovoiman käyttämistä, ja tämän takia pelaajan pitää miettiä, miten käyttää painovoimaa hyväkseen.

Pelissä on myös kaksi muuta peruselementtiä, joista toinen on jo käytössä ja toinen tulee julkaisuversioon sitten, kun pistejärjestelmä on asennettu. Näistä elementeistä ensimmäinen on ”Zone Swaps”, eli painovoimattomasta alueesta painovoimalliseen alueeseen vaihtaminen sekä toisinpäin. Tämä laittaa pelaajan miettimään reittejä, sillä ”Zone Swapsin” loputtua ei voi enää kulkea tilasta toiseen. Kentät ovat yleensä suunniteltu niin, että maali sijaitsee painovoimattomassa tilassa, joten liiallinen kulutus voi pilata maaliin pääsyn.

Toinen tulevaisuudessa tuleva peruselementti on ”Move”, eli liikkuminen pelissä ilman painovoiman hyväksikäyttöä. Tämä on pakollinen painovoimattomassa tilassa ja melkein jokaisessa kentässä tulee olemaan tärkeä myös painovoimallisessa tilassa. Jokaiselle kentälle on määritelty vähimmäisliikkumismäärä, joka määrittelee paljon kentässä saa pisteitä. Pisteytys toimii tähdillä yhdestä kolmeen.

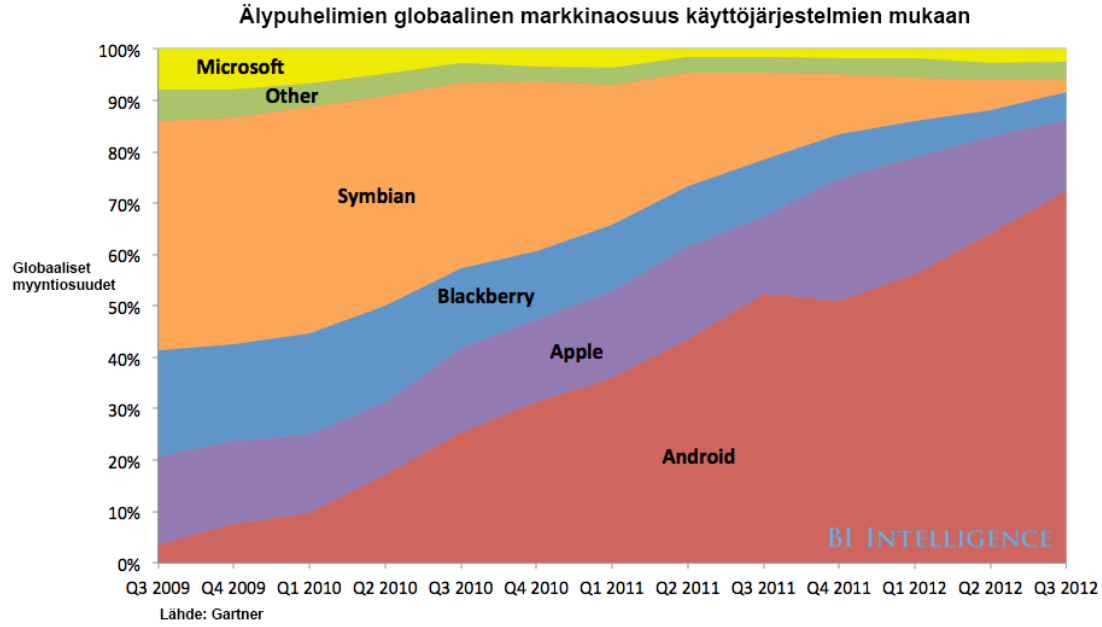
Pelin juoni tulee vasta tulevaisuudessa, sillä sen kehitys on saatu vasta käyntiin. Suurin este juonen ilmisaamiselle on, että se tullaan toteuttamaan graafisella näkymällä ennen kenttiä ja kenttien sisällä.

Konsepti ja juoni tulevat olemaan suunniteltu kaikenikäisille, joten peli ei tule rajaamaan käyttäjiä graafiselta näkymältään. Kuitenkin pelimoottorin ymmärtäminen saattaa tuottaa vaikeuksia lapsille, sillä peli vaatii hieman ongelmanselvittelytaitoja.

4.2.2 Alustan päätös

Suunnitteluvaiheessa tutkittiin, mikä alusta voisi olla kaikista paras pelin suorittamiseen. Niitä oli useita. iPhonen käyttöjärjestelmä eli iOS oli yksi loogisista ratkaisuista, sillä siinä on selvästi parempi kauppajärjestelmä ja turvallisuus kuin Androidilla. Ongelmana oli, että iOS:lle ohjelmointi vaatii XCode-ohjelman, mikäli haluaa rakentaa peliä Applen tukemalla tavalla, ja tämä taas vaatii Applen MacBookin, jota ei työn tekohetkellä ollut saatavilla. Toinen mahdollisuus olisi ollut käyttää valmiita moottoreita, joilla on mahdollisuus tuoda peli iOS-käyttöjärjestelmälle, mutta halusin itse luoda oman pelimoottorin opinnäytetyötä varten.

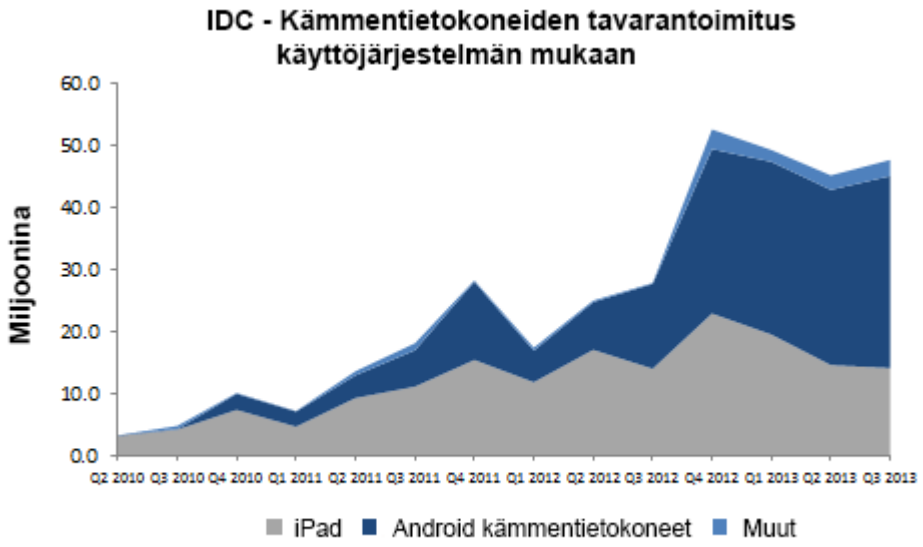
Windows Phonelle pelin tekeminen olisi ollut toinen looginen ratkaisu, mutta ehkä tärkein syy, miksi en tehnyt Windows Phonelle, on markkinaosuus (kuva 6). Markkinaosuudet kertovat, mitkä kilpailevat käyttöjärjestelmät ovat suositumpia sekä kuinka suuri käyttäjäkunta käyttöjärjestelmällä olisi. Tämän takia on hyvä tarkistaa alustojen päätöksessä, mille haluaa tehdä oman sovelluksensa.



Kuva 6. Älypuhelimien markkinaosuus vuosina 2009–2012, jolloin alustan päätös tehtiin. [27.]

Mobiili- ja tablettikehitys eivät olleet ainoita vaihtoehtoja, joille peliä olisi voitu kehittää. Näiden lisäksi suunnitelmissa oli mahdollisesti valita jokin muu alusta kuten Windows, Linux, XBox 360, PS Vita tai alustattomista ratkaisuista HTML5 sekä Flash. Ehkä varteenotettavin näistä vaihtoehdoista oli PS Vita, sillä se on yhtä hyvin suunniteltu pienpeleille kuten mobiililaitteet. Pelin markkinointi PS Vitalle on mahdollista toteuttaa samalla tavalla kuten mobiililaitteilla eli digitaalisella kauppatorilla.

Kaikista vaihtoehdoista kuitenkin yli muiden nousi viime vuonna Android, sillä se loistaa usealla osa-alueella. Androidille pelin ohjelmointi onnistuu Javalla, jota olen opiskellut useamman vuoden koulussa. Androidin markkinaosuus vuonna 2012 oli suuri mobiili- ja tablettimarkkinoilla. Tällä hetkellä viimeisimmän tiedon mukaan sen mobiilimarkkinaosuus on jo ylittänyt 80 % rajan. Samoin Androidin tablettimarkkinaosuus on ohittanut iOS:n ja on uusimpien tuloksien mukaan sen osuus on 65 % kaikista tablettimarkkinoista (kuva 7). [28.]



Kuva 7. Tablettien markkinaosuus vuosina 2010–2013. [29.]

4.3 Toteutus

Pelin toteutus alkoi 2013 keväänä, sekä suurin osa toteutuksesta on rakentunut labyrintti-pelioppaan innoittamana. Oppaassa kehitetään labyrintti, jossa pallo kulkee samalla tavalla kuin omassa pelissäni alkupisteestä loppupisteeseen. Oppaasta oli paljon hyötyä työn aikana. Se muun muassa auttoi havainnoimaan, kuinka Androidissa rakennetaan luokka käyttäen hyväksi Javaa ilman XML-ulkoasua ja kuinka luodaan kenttä sekä lähetetään tiedot pelille. Lisäksi oppaan avulla selvisi, kuinka saadaan Android-alustan mittasuhteet ja käytetään niitä hyväksi palikoiden luomiseen, kuinka piirto toimii pelissä ja miten sitä tulee käyttää pelin päivitykseen muutoksen tapahtuessa. Oppaan vaikutus pelin ohjelmointiin oli suuri, mutta opas ei kuitenkaan ole pelissä avainasemassa, vaan sen lisäksi siihen sovellettiin myös muita ohjeita ja oppaita. [30.]

Peli on suunniteltu täysin geneeriseksi, jonka avulla pelimoottoria kyetään käyttämään muidenkin pelien luontiin. Samalla pelimoottorilla kyetään tekemään hyvin tunnettuja pelejä kuten Tetris tai Bejeweled olisi hyvin helppo luoda käyttäen hyväksi pelin luokkia toteutuksessa. Vaikka peli on suunniteltu yksiajokseksi, niin tämä ei estä millään tavalla luoda moottorille moniajo-käskyjä, joita on mahdollista toteuttaa säikeillä.

Tällä hetkellä peli on ”alpha”-vaiheessa eli pelipakettia jaetaan ainoastaan perheelle ja ystäville, mutta suunnitelmissa olisi saada vuoden 2014 keväänä ensimmäinen julkaisuversio valmiiksi. Pelin ensimmäisen version mekaaninen toteutus on n. 85 %, mistä puuttuvat ainoastaan näkyvät animaatiot, asetukset, tallennusominaisuus, pistejärjestelmä, välivalikko sekä maalivalikko.

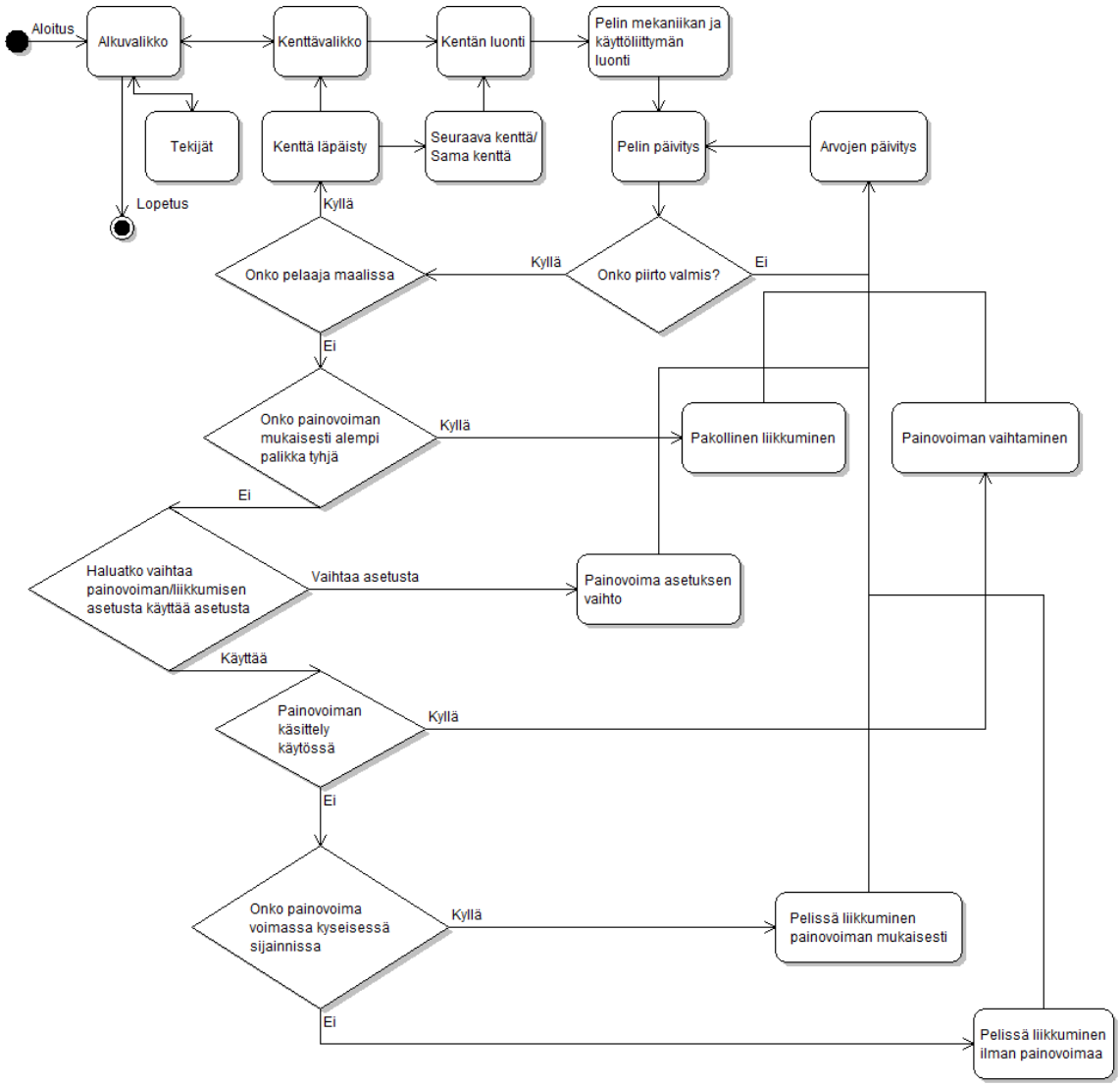
4.3.1 Pelimoottori

Peli on rakennettu 13 Java-luokasta, yhdestä XML-muotoiluluokasta ja kolmesta XML-ominaisuusluokasta (kuva 8). Pelattavan pelin toteutus käyttää hyväkseen näistä ainoastaan kuutta Java-luokkaa ja kahta ominaisuusluokkaa. Aikasemmin pelissä käytettiin useampaa muotoiluluokkaa, mutta koska tämä on helpompi tuottaa geneerisesti Java-luokkien avulla, joten muotoiluluokkaa käytetään ainoastaan pelintekijöiden logon esitykseen avatessa peliä Androidissa.

Kuva 8. Java- ja XML-luokat pelissä

Luokkapyramidi kuvastaa luokkien jaottelua, josta näkyy kuinka peli on luotu alemmista luokista korkeampiin porrasmaisesti. Näistä esimerkiksi TheGame.java luo GameView.java näkymän, joka käyttää hyväksi valmiiksi saatua oliota WorldMaker.java-luokkaa ja tämä luo World.java-olioluokalle arvot, joita GameView-käyttää piirrossa.

Pelin aloitus toimii MainMenu-luokassa, jossa on mahdollista valita kolmesta käskystä, mitä tehdään: aloitetaan kenttävalikko, luetaan tekijöistä tai lopetetaan peli. Pelin lopetus suinkaan ei tarkoita ohjelman sammumista, vaan Android asettaa pelin taustalle ja lopettaa sen itse automaattisesti, mikäli tarvitsee muistipaikkaa kyseisen ohjelman tilalle. Tekijä-aktiviteetti sisältää ainoastaan tietoa tekijöistä, josta takaisin-napin avulla pääsee takaisin MainMenu-luokkaan. Kenttävalikoima sisältää kentät ja siinä voi valita, mitä kenttää haluaa pelata. Kun kenttä on valittu, luo tämä halutun kentän, alustaa mekaniikan ja luo kentälle käyttöliittymän (kuva 9).



Kuva 9. Pelin mekaniikan prosessikaavio

Varsinaisesta pelistä vastaavat seuraavat luokat: GameView.java, Sound.java, TheGame.java, Tile.java, World.java ja WorldMaker.java. Näistä TheGame.java on aktivi-

teettiluokka, GameView.java on näkymäluokka, Tile.java ja World-luokka ovat olioluokkia sekä WorldMaker.java ja Sound.java staattisia luokkia.

TheGame.java-luokan toteutus on hyvin yksinkertainen. Se on perustana sille, että varsinainen peli toimii, mutta itse luokassa TheGame.java delegoi GameView.java luokalle tehtävän, että tässä on kenttä, toteuta kaikki muut (koodiesimerkki 1).

```
Intent intent = getIntent();  // Aikomuksen luonti ja WorldSelectionista saadut tiedot
Bundle extras = intent.getExtras(); // Kerätään WorldSelectionista antamat tiedot
World world = (World)extras.get("world"); // Maailman luonti, joka luodaan tietojen saatujen mukaisesti
GameView view = new GameView(this, world); // Luodaan käyttöliittymälle näkymä ja lähetetään konteksi + kenttä
setContentView(view); // Asetetaan näkymä käyttöön
```

Koodiesimerkki 1. TheGame.javan ohjelmallinen aktiviteettitoteutus

GameView.java -luokka ottaa käskyn vastaan ja aloittaa varsinaisen toteutuksen. Ensimmäisenä GameView.java luo pelille mekaniikan ja käyttöliittymän perustuen käytettävän Androidin omiin mittasuhteisiin. Mittasuhteet eivät ole aina samoja, joten pelin pitää ymmärtää luoda mittasuhteita dynaamisesti riippuen laitteen resoluutiosta. Toinen asia, mikä pelin pitää ymmärtää, on mittasuhteiden sovittaminen kentän kokoon. Kolmas elementti, joka rajoittaa kentän kokoa mittasuhteilla, on topBar, joka on yläpalkki pelissä. topBar:iin on sijoitettu ”Zone Swaps” ja ”Move” laskurit (koodiesimerkki 2).

```
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
    width = w; // Alustan leveys
    height = h; // Alustan korkeus
    topBar = (height*topBarSize); // Yläpalkki
    topBarBorder = (topBar*topBarBorderSize);
    topBarBlock = (topBar-topBarBorder);
    gameHeight = height - topBar; // Kentän korkeus
    cellHeight = gameHeight / sizeY; // Palikan korkeus
    cellWidth = cellHeight; // Palikan leveys (rakennettu neliöksi)
    gameWidth = cellWidth * sizeX; // Kentän leveys
    world.setGameWidth(gameWidth); // Lähetä kentälle sen leveys
    world.setGameHeight(gameHeight); // Lähetä kentälle sen korkeus
    super.onSizeChanged(w, h, oldw, oldh);
}
```

Koodiesimerkki 2. Geneeriset mittasuhdetoteutukset resoluutiosta riippumatta.

Osa alustusta sekä pelin toimintaa on piirtometodi. Pelissä java-pohjaiset muotoilut on rakennettu Canvas-elementin avulla. Tätä elementtiä käytetään piirto-metodissa, joka rakentaa pelille graafisen käyttöliittymän. Canvas-elementtien metodit eivät ole kaikki samanlaisia, mutta periaate on usein samankaltainen. Esimerkiksi jokaisen kentän seinäpalikoiden piirtäminen on: ”canvas.drawBitmap(kuva, vasen reuna, yläreuna, väritys)” (koodiesimerkki 3).

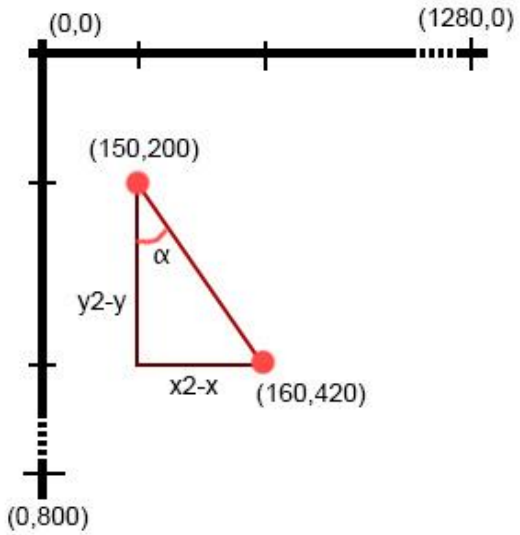
```
// Kuvan hakeminen ja skaalaus yhden palikan kooksi + 1px, suoritettu aikasemmassa vaiheessa.
wallBlock = BitmapFactory.decodeResource(getResources(), R.drawable.wallblock);
wallBlockS = Bitmap.createScaledBitmap(wallBlock, (int)cellWidth+1, (int)cellHeight+1, true);
// Värityksen luonti ja asetukset, luotu alustuksen yhteydessä
bgimg = new Paint(); // Värin luonti
bgimg.setAntiAlias(true);  // Pehmentää reunoja
bgimg.setFilterBitmap(true); // Filtteröi kuvaa
bgimg.setDither(true); // Lisää väritarkkuutta
// Käy läpi kaikki palikat x,y tasossa
for(j = 0; j < sizeY; j++) {
    for (i = 0; i < sizeX; i++) {
        if(world.tileType == 3) { // Mikäli kyseessä on seinä
            canvas.drawBitmap(wallBlockS, // Skaalattu kuva
            ((width-gameWidth)/2)+(cellWidth*i), // vasen reuna
            topBar+(cellHeight*j), // oikea reuna
            bgimg); // väritysasetukset
        }
    }
}
```

Koodiesimerkki 3. Seinäpalikoiden piirtäminen for-silmukassa ja palikan if-tarkistuksen sisällä.

Alustuksen jälkeen peli toimii odottaen seuraavaa käskyä, mikä kosketusnäytöllisellä mobiili- tai tabletti-Android-laitteilla tarkoittaa näyttöön koskemista. Peli on rakennettu niin, ettei näytön kosketuskohdalla ole väliä, vaan ainoastaan sillä, minkälainen kosketus on kyseessä. Peli vaatii pelaajaa tekemään joko yhden tai kahden sormen kosketuksen. Yhden sormen kosketuksesta peli laskee sormen kosketuskohdan ja sormen nostokohdan etäisyyden ja asteen. Tämän laskettuaan peli päättää, oliko kyseinen kosketus sopivassa vaaka- tai pystytasossa niin, että kosketus oli halutunlainen.

Esimerkiksi: (x,y) on alkupiste ja (x2,y2) on loppupiste. Peli haluaa kosketuksen +-30-asteen kulmassa, pituus on 10 % pystytasolla tai 6 % vaakatasolla koosta 1280x800 resoluutiollisella näytöllä. Käyttäjän alkukosketus on (150,200) ja kosketus loppuu pisteessä (160,420).

Peli ajattelee tätä esimerkkiä niin, että mihin suuntaan kosketus siirtyi laskemalla erotukset x2-x: 160-150=10 ja y2-y: 420-200=220. Tästä voidaan päätellä, että kosketus on siirtynyt alaspäin, sillä peli rakentuu ainoastaan 0-1280 x-akselille ja 0-800 y-akselille (kuva 10).



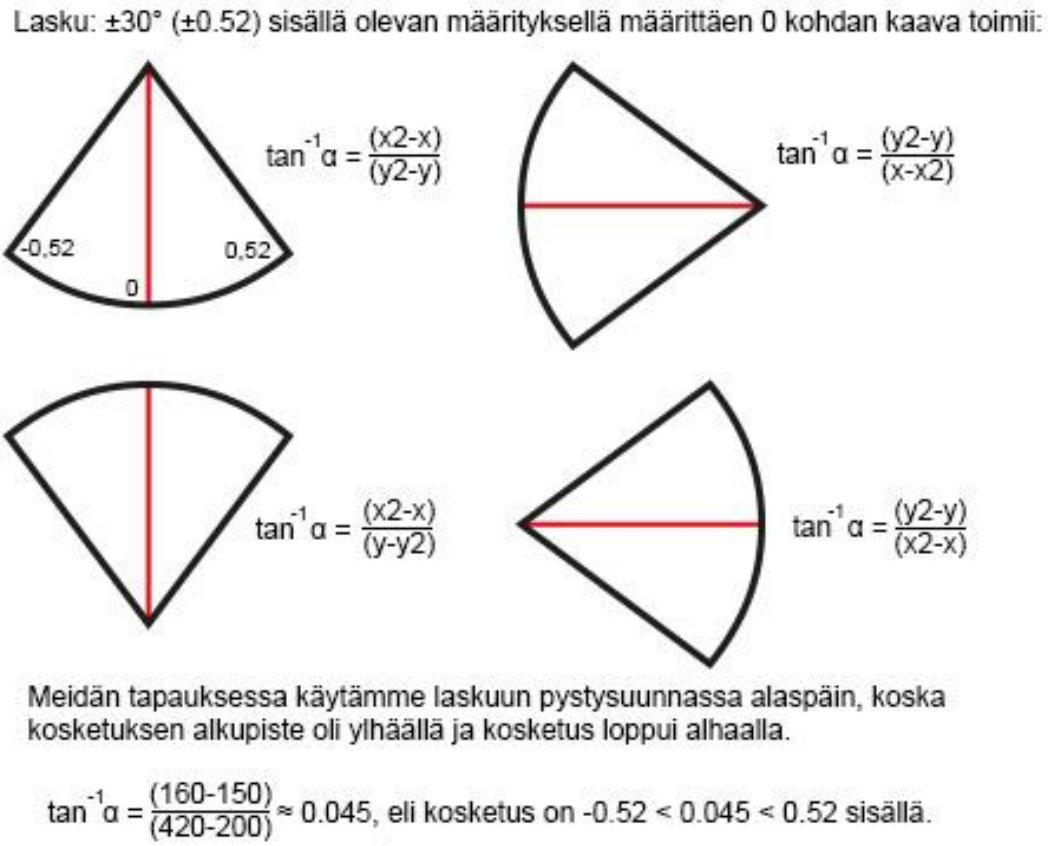
Kuva 10. Pisteiden geometrinen määritys ja asteen laskenta.

Kosketus on pystytasossa 220 ja vaakatasossa 10. Näistä on vaadittu 80 pystytasossa tai 76,8 vaakatasossa kosketuksen pituudeksi. Ainoastaan pystytason siirto oli tarpeeksi pitkä, joten nollapiste asetetaan positiiviseen pystytasoon nähden alaspäin. Tämän jälkeen on asteen laskemisen vuoro. Ensimmäisenä muutetaan +-30-asteen kulma radiaaneiksi (kuva 11).

$$30° = \frac{\pi}{6} \approx 0.52 \quad \text{Eli} \pm 30° = \pm 0.52$$

Kuva 11. +-30-asteen muuttaminen +-radiaaneiksi.

Määrittäessä radiaanin nollakulmaa, täytyy lähteä siitä, että millä laskutavalla tätä aiotaan laskea. Laskutapa on yksinkertainen sekä tarvittavat osat ovat alpha-kulma, viereinen kateetti ja vastainen kateetti (kuva 12). [31.]



Kuva 12. Alpha-kulman määrittäminen radiaaneina.

Koska kosketus oli radiaaneina halutun tuloksen sisällä, ohjelma suorittaa halutun käskyn. Käskyn tapa voi tarkoittaa kahta asiaa: Säädetäänkö painovoiman suuntaa vai liikkuuko hahmo riippuen hahmon sijainnista (koodiesimerkki4) (koodiesimerkki 5).

```
// Alas (SOUTH)
if((y2-y) >= (gameHeight*0.1)) { // Pituuden tarkistus
    rad = Math.atan2((x2-x), (y2-y)); // tan^-1 = (x2-x)/(y2-y)
    if(rad<=0.52 && rad>=-0.52){ // radiaani välillä +-30-astetta
        if (gravity != NORTH) { // Mikäli painovoima ei ole pohjoinen
            if (gravity != SOUTH) { // tai etelässä
                gravity = SOUTH; // Painovoiman säätö etelään
                moved = true; // Komento on tosi
            }
        }
    }
}
```

Koodiesimerkki 4. Painovoiman säätäminen etelään, eli alaspäin pelissä. [32.]

```
// Kosketus alaspäin
if((y2-y) >= (gameHeight*0.1)) { // Pituuden tarkistus
    rad = Math.atan2((x2-x), (y2-y)); // tan^-1 = (x2-x)/(y2-y)
    if(rad<=0.52 && rad>=-0.52){ // radiaani välillä +-30-astetta
        if (tileType == 2) { // Mikäli palikka on painovoimallinen
            moved = movePlayerGravity(DOWN); // Liikuta hahmo alas
        } else if (tileType == 1) { // Mikäli palikka on painovoimaton
            moved = movePlayerFreely(DOWN); // Liikuta hahmo alas
        }
    }
}
```

Koodiesimerkki 5. Hahmon liikkuminen alaspäin, riippuen onko painovoima voimassa kyseisessä kohdassa. [32.]

Toinen kosketustapa on käyttää kahden sormen kosketusta. Kahdella sormella näyttöön koskeminen laukaisee painovoiman säätöasetuksen joko päälle tai pois päältä. Mikäli painovoiman säätöasetukseen kosketaan, sen tulee myös asettaa sormen nosto-ominaisuus pois päältä, koska muuten se prosessoisi vahingossa myös yhden sormen kosketuskäskyn (koodiesimerkki 6). [33.]

```
case MotionEvent.ACTION_POINTER_UP: // Toinen kosketus näyttöön
    blockSecondEvent = true; // Estä yhden sormen kosketuskäsky
    if (gravityTouch == false) { // Mikäli painovoiman säätö pois käytöstä
        gravityTouch = true; // Laita painovoiman säätö käyttöön
        moved = true; // Komento on tosi
    } else {
        gravityTouch = false; // Laita painovoiman säätö pois käytöstä
        moved = true; // Komento on tosi
    }
    refresh(moved); // Päivitä piirto-metodin avulla
    break;
```

Koodiesimerkki 6. Kahden sormen kosketus

Kummatkin kosketuskäskyt toimivat saman refresh (tosi/epätosi) -päivitysmetodin alaisena. Jos refresh-metodin lähetettävä arvo on tosi, ohjelma ymmärtää jonkin muuttujan vaihtaneen arvoa, jolloin sen tulee päivittää piirto. Mikäli lähetettävä arvo on epätosi, ohjelma lähettää äänen, ettei kosketus ollut pelimekaniikan mukainen siirto (koodiesimerkki 7). [34.]

```
if (gravityTouch == false) { // Mikäli painovoiman säätö pois käytöstä
    moved = world.touchMove(touchX1, touchY1, touchX2, touchY2); // Yhden sormen liikkuminen
    if (moved == false) { // Liike ei ollut pelimekaniikalle sopiva
        Sound.play(context, R.raw.dup); // Lähetä ääni: "dup"
    } else {
        refresh(moved); // Päivitä piirto-metodin avulla
    }
} else {
        moved = world.touchGravityMove(touchX1, touchY1, touchX2, touchY2); // Yhden sormen painovoiman vaihto
    if (moved == false) { // Liike ei ollut pelimekaniikalle sopiva
        Sound.play(context, R.raw.dup); // Lähetä ääni: "dup"
    } else {
        refresh(moved); // Päivitä piirto-metodin avulla
    }
}
```

Koodiesimerkki 7. Päivitys-metodin määrittely kosketuksen perusteella

Kun päivitysmetodi saa käskyn, ensimmäisenä se päivittää piirtonäkymän, jonka jälkeen se tarkistaa, onko piirrossa hahmo siirtynyt alkupalikasta seuraavaan palikkaan. Tämä ominaisuus on osa animaatiota, jolla on luotu hahmon siirtyminen rekursiivisella tavalla viidesti, ennen kuin ollaan seuraavassa palikassa. Rekursiivinen tapa toimii luomalla metodin sisällä uuden käskyn suorittaa samaa metodia, jonka sisällä käsky tehdään.

Rekursiivisen animaation jälkeen hahmo on siirtynyt haluttuun palikkaan, jonka jälkeen tarkistetaan, onko hahmo päässyt maaliin. Mikäli hahmo ei ole vielä maalissa, metodi jatkaa päivittämistä tarkistaen, onko hahmo painovoimallisessa tilassa ja onko painovoiman mukaisesti alempi palikka tyhjä. Näin hahmo ei voi jäädä painovoimallisessa tilassa leijumaan ilmaan, mikä myös toimii rekursiivisella tavalla (koodiesimerkki 8).

```
1  public void refresh(boolean moved) {
2      invalidate(); // Piirto päivitys
3      moved = drawCheck(); // Tarkista onko piirto valmis
4      if(moved) { // Rekursiivinen rakenne, mikäli piirto ei ole valmis
5          refresh(moved);
6      } else {
7          if(world.isGameFinished() == true) {
8              ... // Sisältää mitä tapahtuu, kun pääsee maaliin.
9          }
10         moved = gravityCheck(); // Painovoimallisen sijainnin tarkistus
11         if(moved) { // Rekursiivinen rakenne, mikäli hahmo leijuu ilmassa
12             refresh(moved);
13         }
14     }
15 }
```

Koodiesimerkki 8. Päivitysmetodi: rekursiiviset käskyt ja maaliin pääsy.

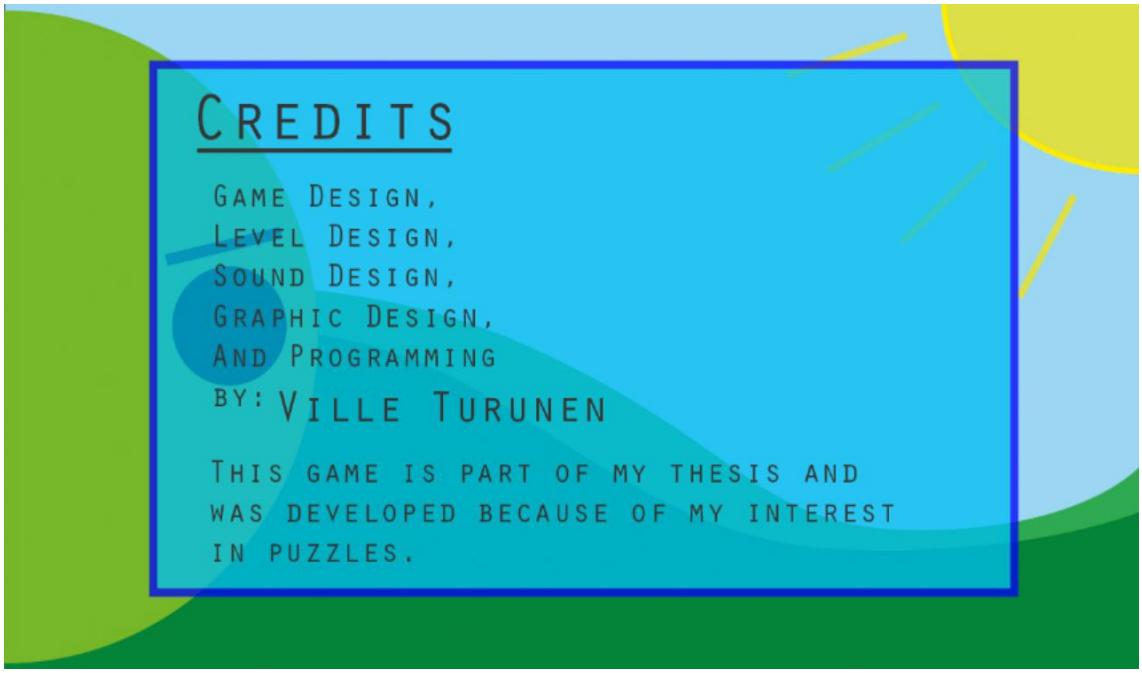
Päivityksien jälkeen peli palaa odotustilaan, jossa se odottaa seuraavaa siirtoa eli käyttäjän kosketusta kosketusnäyttöön.

4.3.2 Pelin käyttöliittymä

Pelin käyttöliittymä koostuu viidestä osasta: alkuvalikosta, tekijästä, kenttävalikosta, pelinäkymästä ja loppunäkymästä. Pelin käynnistyessä ensimmäinen näkymä on alkuvalikko (kuva 13).

Kuva 13. Alkuvalikko-näkymä

Kun alkuvalikossa kosketetaan kohtaa ”Credentials” eli tekijät, tulee tekijät-näkymä esille. Tekijät-näkymä ei sisällä lainkaan painikkeita, joten tästä pois pääseminen vaatii Androidin omaa takaisin-painiketta (kuva 14).

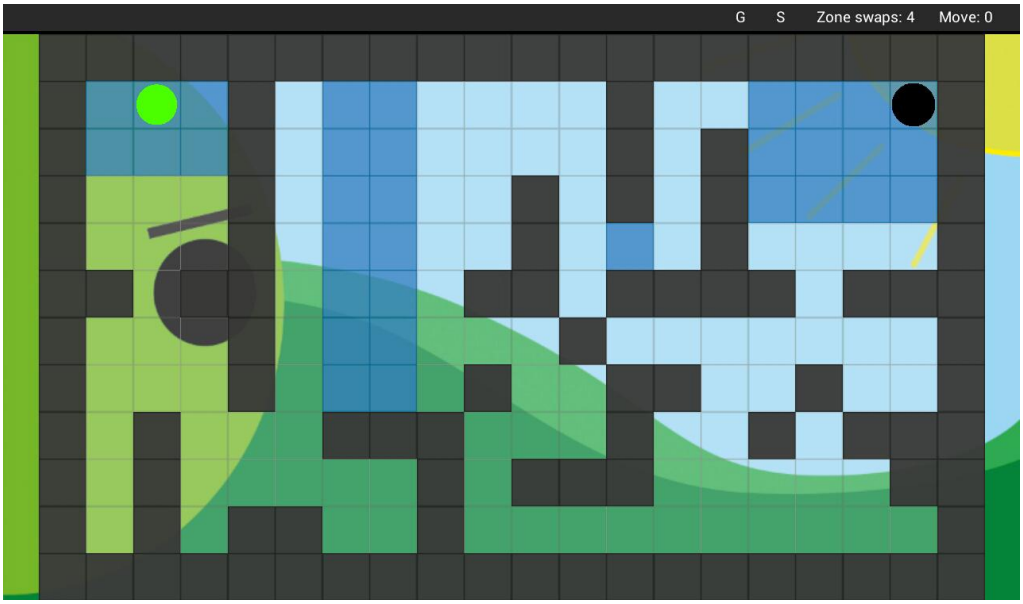


Kuva 14. Tekijät-näkymä

Kun alkuvalikossa kosketetaan kohtaa ”Start” eli kenttävalikoiman valitsemista, tuodaan esille kenttävalikko-näkymä. Kenttävalikko on rakennettu Java-pohjaisesti. Se on kehitetty niin, että käyttäjä pääsee kentissä eteenpäin ainoastaan, kun nämä ovat valmiita ja aikaisemmat kentät ovat hyväksytysti läpäisty. Ensimmäisessä julkaisuversiossa kenttiä tulee olemaan viisi opaskenttää ja 3*25 tasokenttää. Tällä hetkellä kenttiä on vain pari kappaletta pelissä ja yli kymmenen kenttää suunniteltuna (kuva 15).

Kuva 15. Kenttävalikko-näkymä

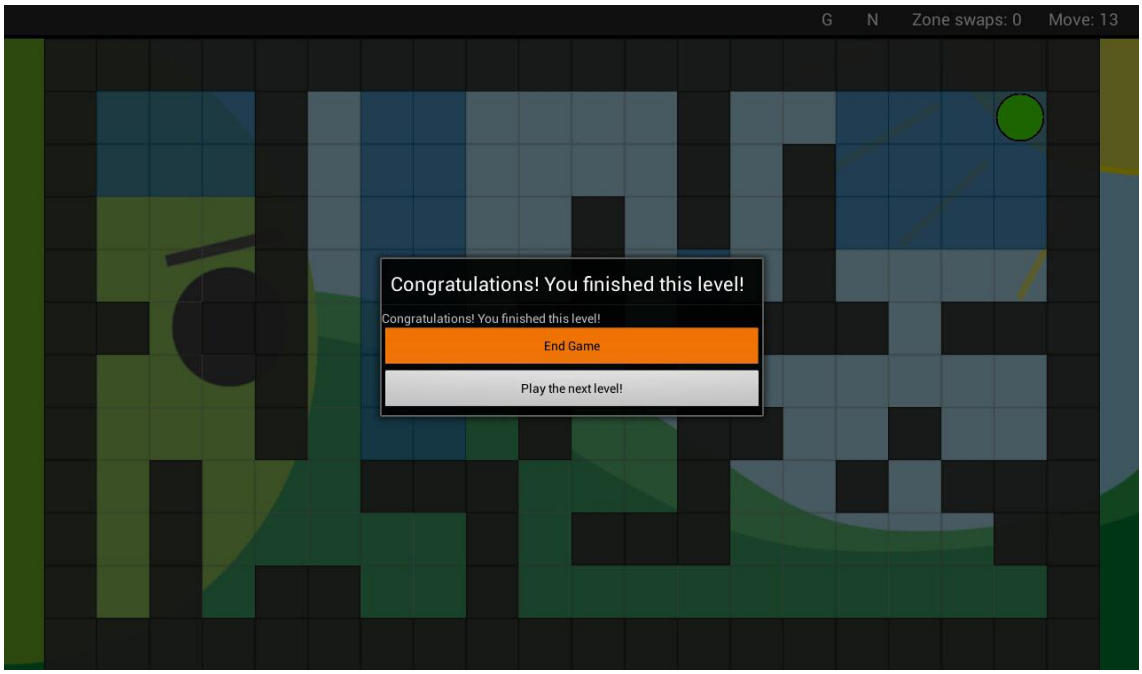
Kun kenttä on valittu, alkaa itse pelin varsinainen käyttöliittymä eli pelinäkymä. Tämä on pelissä kaikista tärkein näkymä, koska näkymä sisältää itse pelin idean. Pelinäkymä on rakennettu kolmesta eri osasta: yläpalkista, taustakuvasta ja pelialueesta. Yläpalkissa sijaitsevat aikaisemmin jo läpikäydyt kaksi peruselementtiä: ”Zone swaps” ja ”Move” sekä tämän lisäksi kaksi muuta elementtiä: painovoiman suunta ja painovoiman säädin (kuva 16).



Kuva 16. Pelinäkymä

Pelinäkymä ei ole vielä pysyvä, sillä se tulee muuttumaan tulevaisuudessa. Yläpalkkiin tulee muutoksena painovoiman ja painovoiman säätimen yhdistäminen yhdeksi kuvaksi, joka kertoo suunnan ja värin avulla onko se voimassa. Pelinäkymän muita tulevia muutoksia ovat grafiikoiden uudistus, taustakuvan vaihtaminen teemoittain kentän mukaisesti sekä asetukset-painike, jolla pääsee välivalikkoon vaihtamaan pelin asetuksia.

Viimeisin näkymä on loppunäkymä, joka ilmaantuu pelinäkymän päälle, kun pelaaja pääsee maaliin. Loppunäkymä on rakennettu XML-pohjaisesti ja on toiminnaltaan hyvin yksinkertainen. Sen avulla käyttäjä voi valita, haluaako lopettaa pelin vai jatkaa seuraavaan kenttään. Loppunäkymä tulee muuttumaan ennen julkaisua, jonka jälkeen pelaaja näkee pisteytyksensä. Tämän lisäksi pelaaja voi valita haluaako pelata saman kentän uudelleen, mennä seuraavaan kenttään vai mennä kenttävalikkoon (kuva 17).



Kuva 17. Loppunäkymä

4.3.3 Grafiikka

Peli sisältää tällä hetkellä yhteensä 20 kuvaa. Näistä suurin osa pienistä kuvista on standardoituja 64x64 pikselikoolle ja taustakuvat taas 1280x720 pikselikoolle. Kuvat pelissä eivät kuitenkaan noudata pelkästään näitä kokoja, vaan ne ovat dynaamisesti skaalauntuvia tarpeen mukaan. Tämä mahdollistaa geneerisen toteutuksen, että pelaaja voi pelata pienellä tai isolla näytön tarkkuudella.

Kuvat on luotu käyttäen hyväksi PNG-kuvaformaatin sisäisiä ominaisuuksia. Tämä kuvaformaatti sisältää ominaisuuden, että kuvasta voi tehdä jo tekovaiheessa läpinäkyvän, joka automaattisesti toimii myös pelissä. Toinen hyödyllinen ominaisuus verrattuna joihinkin muihin kuvaformaatteihin on kuvan tarkkuus. Tämä on PNG:llä parempi kuin esimerkiksi JPG-kuvaformaatisissa, jonka takia kuva pysyy tarkkana skaalatessa sitä isommaksi tai pienemmäksi. Ainoa huonompi puoli muihin kuvaformaatteihin verrattuna on kuvan suuri tiedostokoko, mutta näin yksinkertaiselle pelille tämä ei tuota ongelmia.

4.3.4 Äänet

Pelissä ovat käytössä tällä hetkellä ainoastaan yksittäiset äänet. Peliin olisi mahdollista luoda musiikki taustalle, mutta mobiilipelejä pelataan usein julkisilla paikoilla, jonka takia tämä ei ole välttämätön ominaisuus. Pelissä on luotu oma staattinen luokka, joka käyttää ääniominaisuutta play- ja stop-metodeilla. Stop-metodia ei käytetä juurikaan itse, koska äänet ovat hyvin lyhyitä, eikä ole tarpeen pysäyttää ääntä kesken suorituksen. Aina play-metodia käytettäessä tämä varmistaa, että ääni alkaa alusta ja käyttää stop-metodia varmistaakseen, ettei ääni ole sillä hetkellä käytössä (koodiesimerkki 9). [34.]

```
1   // Lopeta vanha ääni ja soita uusi
2   public static void play(Context context, int resource)
3   {
4       stop(context); // Pysäytä-metodi
5       mp = MediaPlayer.create(context, resource); // Luo uusi ääni
6       mp.setLooping(false); // Varmista ettei ääni pyöri silmukassa
7       mp.start(); // Soita ääni
8   }
9   // Pysäytä ääni
10  public static void stop(Context context)
11  {
12      if (mp != null)
13      {
14          mp.stop(); // Mikäli ääni on käytössä niin pysäytä tämä
15          mp.release(); // Vapauta äänirauta
16          mp = null; // tyhjennä äänimuuttuja
17      }
18  }
```

Koodiesimerkki 9. Play- ja Stop-metodien toiminta [34.]

4.3.5 Kenttäsuunnittelu

Kenttäsuunnittelussa käytettiin hyväksi omaa kenttärakennusluokkaa, WorldMaker.javaa. Kenttävalikossa valitaan kenttä, jota halutaan pelata, jonka jälkeen kentän rakentaja tarkistaa, mikä on kentän numero ja alustaa world-luokan kentän ominaisuuksilla. Jokainen kenttä tarvitsee neljä eri kohtaa, joilla se erottuu muista kentistä: kentän taulukko, aloituspiste, maalipiste ja ”Zone swaps”.

Kenttä on suunniteltu niin, ettei pelkästään yksi taulukko riitä kertomaan, miten kenttä rakennetaan, vaan tämä tarvitsee myös vaaka- ja pystytason sijainnin. Tämän mahdollistamiseksi tarvitaan uusi olio-luokka: Tile.java. Kentän rakentamisen aikana World-luokan setTile-metodille lähetetään tietona x, y ja palikan tyyppi. Tämän jälkeen World-luokka luo uuden olion Tile-luokasta omaan taulukkomuuttujaansa, joka toimii Tile-oliona (koodiesimerkki 10).

```
// WorldMaker-luokan sisältö:
int[][] blocks = new int[][] {
    {3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3},
    {3, 1, 1, 1, 3, 2, 1, 1, 2, 2, 2, 2, 3, 2, 2, 1, 1, 1, 1, 3},
    {3, 1, 1, 1, 3, 2, 1, 1, 2, 2, 2, 2, 3, 2, 3, 1, 1, 1, 1, 3},
    {3, 2, 2, 2, 3, 2, 1, 1, 2, 2, 3, 2, 3, 2, 3, 1, 1, 1, 1, 3},
    {3, 2, 2, 2, 3, 2, 1, 1, 2, 2, 3, 2, 1, 2, 3, 2, 2, 2, 2, 3},
    {3, 3, 2, 3, 3, 2, 1, 1, 2, 3, 3, 2, 3, 3, 3, 3, 2, 3, 3, 3},
    {3, 2, 2, 2, 3, 2, 1, 1, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 3},
    {3, 2, 2, 2, 3, 2, 1, 1, 2, 3, 2, 2, 3, 3, 2, 2, 3, 2, 2, 3},
    {3, 2, 3, 2, 2, 2, 3, 3, 3, 2, 2, 2, 3, 2, 2, 3, 2, 3, 3, 3},
    {3, 2, 3, 2, 2, 2, 2, 2, 3, 2, 3, 3, 3, 2, 2, 2, 2, 2, 3, 3},
    {3, 2, 3, 2, 3, 3, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3},
    {3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3}
};
sizeX = blocks[0].length; // Vaakatason suuruus
sizeY = blocks.length; // Pystytason suuruus
world.setSizeX(sizeX); // Aseta kooksi sizeX
world.setSizeY(sizeY); // Aseta kooksi sizeY
for(j = 0; j < sizeY; j++) { // Silmukassa arvojen lähettäminen
    for (i = 0; i < sizeX; i++) {
        world.setTile(i, j, blocks[j][i]); // setTile-metodille tiedot
    }
}
world.setSwapCount(4); // "Zone Swaps"
// World-luokan sisältö:
public void setTile(int x, int y, int block) {
    tile[x][y] = new Tile(x, y, block); // Luo olioita valmiiseen tile-taulukkoon
}
```

Koodiesimerkki 10. Kentän luonti WorldMaker- ja World-luokissa.

Alkupisteen ja maalin luonti eivät sisälly taulukkoon, koska ne voivat olla missä tahansa palikassa. Tämän takia näille tulee antaa omat pisteensä, jotka toimivat sallitun pelikentän sisällä. Tässä vaiheessa on tärkeää, että kentän tekijä tarkistaa mihin laittaa

aloitus- ja loppupisteen, sillä peli ei estä laittamasta niitä rajojen ulkopuolelle tai seinän päälle. Vielä kaikenlisäksi pitää huomioida palikoiden alkavan 0-pisteestä, joten (0,0) on ensimmäinen kohta, eikä (1,1) (koodiesimerkki 11).

```
1    world.setStart(2, 1); // Aloituspiste
2    world.setGoal(18, 1); // Maali
```

Koodiesimerkki 11. Aloitus- ja loppupisteen määritys.

4.4 Testaus ja ohjelmointivirheiden kirjaus

Testaamisen pois jättäminen on ehkä pahin virhe, minkä voi tehdä sovelluskehityksessä. Peliä ei pidä ajatella siltä kannalta, että kunhan se toimii, niin kaikki on kunnossa, vaikka pelissä olisikin pari ohjelmointivirhettä. Tämä voi kostautua jälkeenpäin erittäin vakavilla seurauksilla, koska pienikin virhe voi muuttaa koko pelimekaniikan pohjustaa, mikäli sitä ei ole mahdollista korjata millään muulla tavalla. Pelissä testaus on ollut jatkuvassa integroinnissa ohjelmoinnin kanssa, jolloin jokainen pieni virhe on korjattu välittömästi tai laitettu ylös, jotta se korjattaisiin ennen kehityksen jatkamista. Pienissä peleissä tämä on hyvä tyyli ja sopii testaamiseen paremman puutteessa, mutta kyseinen malli ei välttämättä toimi isommassa projektissa (liite 2).

Testauksen perustaa tulisi ajatella niin, että indie-peliä ei pitäisi julkaista ennen kuin on varma, ettei ole virheitä pelissä, jotka pilaavat pelaajan elämyksen. Tämä tietenkin on riippuvainen myös muista tekijöistä kuten esimerkiksi siitä, onko koko pelin ohjelmoinut sama henkilö tai onko käytössä jokin ulkoinen rajapinta. Tämä perustuu siihen, että mikäli ohjelmoija ei tunne pelinsä toimintaa, niin peli ei välttämättä toimi sillä tavalla kuin sen pitäisi. Jokainen ohjelmointivirhe saattaa pilata pelin elämyksen, eikä tämä ole sitä, mitä pelin tekijä haluaa pelaajille luoda.

Virheiden kirjaaminen on haastava puuha, koska jos peliä tekee useampi ihminen, joista joku löytää virheen, täytyy hänen kyetä kirjoittamaan tai sanomaan virhe selvästi, jotta sen korjaaja ymmärtää, mistä puhutaan. Useammat yritykset panostavat virheiden korjaamiseen hankkimalla virheiden ja tukipyyntöjen raportointiohjelmia. Tällaisia ovat esimerkiksi JIRA, wiki tai Bugzilla. [35.]

Indie-peliohjelmoinnissa ei välttämättä tarvita saman luokan ohjelmointivirheiden kirjaamista, kuin mitä normaaleissa yrityksissä käytetään. Virheet voi hyvin kirjata normaaleihin tekstitiedostoihin, kunhan pitää kirjaa siitä, missä vaiheessa virheen korjaus on tai jos se on jo suoritettu. (Liite 2)

4.5 Julkaisu ja markkinoinnin osat

Pelin julkaisu tulee tapahtumaan 2014 keväällä, koska peli ei tule vielä keskeneräisenä vapaaseen julkaisuun. Julkaisu tulee tapahtumaan alustavasti Google Play -kauppaan, jonka jälkeen riippuen pelin menestyksestä sen kehitystä jatketaan muihin jakelukanaviin. Julkaisussa tulee olemaan kaksi versiota: ilmainen ja maksullinen versio. Näiden ero ei ole millään tavalla merkittävä, koska pelin on tarkoitus kerätä mahdollisimman paljon lataajia, eikä tuottoa.

Maksullinen versio on tarkoitettu tukipakettina tekijälle pelistä. Myös päivitykset tulevat ensimmäisenä maksulliseen versioon, jonka jälkeen parin viikon viiveellä ne tulevat myös ilmaiseen versioon. Toinen hyöty maksullisessa versiossa on lupaus seuraaviin peleihin, jonka avulla aikaisemmin tukeneet saavat mainokset pois tulevista peleistä oman käyttäjäprofiilin avulla.

Markkinointia suunnitellessa on hyvä varautua siihen, kuinka paljon on käyttäjiä sillä alustalla tai alustoilla, joille aikoo julkaista oman sovelluksensa. Suuren käyttäjämäärän saamiseksi tekijän tulisi miettiä pikemminkin sitä, kuinka aikoo erottua joukosta, kuin esimerkiksi ikäryhmää, jolle sovellus on tarkoitettu. Mitä vähemmän käyttäjiä sovelluksen alustalla on, sitä tärkeämpää sovellukselle on kohdeyleisön valinta sekä mahdollisen markkinaraon kartoitus.

Yleisesti ajatellaan, että markkinointi maksaa paljon ja että on vaikeaa saada minkäänlaista näkyvyyttä ilman rahallista apua. Tämä on yksi asia, mikä on muuttunut vanhasta mallista, sillä nykyään on monta toimivaa promotointitapaa, joita voidaan hyödyntää ilmaiseksi tai vähäisellä rahalla.

Ennen markkinointia pitää varautua kysymykseen: ”Mitä markkinoin?” Onko se peli, tunnus, brändi, video tai jokin muu? Pelkästään pelistä voidaan promotoida kaikkia

näistä. Hyvän markkinoinnin perusteena on, että pystyy tuomaan pelin näkyvyyden esille usealta eri kannalta, mikä taas innostaa pelaajia lataamaan tai ostamaan pelin.

Pelin pelaamista ajatellaan elämyksenä tai kokemuksena, eikä niinkään tuotteena. Tämän takia paperilta kuvan näkeminen ei kerro, minkä elämyksen henkilö voisi saada pelatessa. Jotta tästä saataisiin mahdollisimman samankaltainen kokemus, tulee kuvan kertoa henkilölle nopeasti, mikä pelistä tekee mielenkiintoisen.

Pelin promotointi tulee tapahtumaan sosiaalisessa mediassa, videopalveluissa, pelin tekijän kotisivuilla sekä keskustelupalstoilla. Mahdollisia lisäpromootioväyliä ovat lehdistö, mainoslehtiset LAN-tapahtumissa ja kilpailut.

4.6 Jatkosuunnitelmat ja ylläpito

Pelin rakentaminen tulee jatkumaan ensimmäisestä julkaisusta vähintään siihen asti, kunnes peli sisältää kenttiä 180 kappaletta. Tämän lisäksi pelissä tulevat kehittymään sen pelimekaniikka, grafiikka ja äänet. Peliä tullaan ylläpitämään myös lopullisen valmistumisen jälkeen päivittämällä se toimivaksi uudemmilla Android-versioilla. Myös mahdolliset ohjelmistovirheet korjataan.

Peli ei välttämättä tule jäämään pelkästään Android-alustalle. Jatkosuunnitelmat riippuvat siitä, kuinka pelaajakunta kehittyy ja onko kysyntää rakentaa peliä muille alustoille.

5 Indie-pelinkehittäjien mielipidetiedustelu

Opinnäytetyötä varten tehtiin kolmessa suljetussa sosiaalisen median ryhmässä avoin mielipidetiedustelu indie-pelinkehittäjien menestyksestä, pelin rakentamisesta ja heidän mielipiteistään. Mielipidetiedustelu tehtiin 13.11.2013 – 14.11.2013 ja siihen saatiin yhteensä kymmenen vastaajaa.

5.1 Mielipidetiedustelun suunnittelu ja toteutus

Mielipidetiedustelu suunniteltiin tukemaan opinnäytetyössä käsiteltyjä aiheita liittyen Indie-pelinkehitykseen sekä saamaan uusia näkökulmia siitä, kuinka muut ovat alalla

pärjänneet. Toteutus tehtiin Googlen lomakeominaisuudella, jonka avulla kyselyn levittäminen oli yksinkertaista ja vastaukset pystyi lukemaan ainoastaan lomakkeen tekijä.

Tiedusteluun oli laitettu 14 kohtaa, joista kysymyksiä oli 11 ja pakollisia kohtia viisi. Pakolliset kysymykset liittyivät vastaajaan sekä hänen tekemiinsä peleihin. Lähes kaikki vastaajat vastasivat kaikkiin tarpeellisiin kysymyksiin. Tuloksissa tullaan käsittelemään ainoastaan 4 - 12 kysymykset, sillä 1 - 3 ja 13 – 14 liittyivät taustoihin ja kysymyksen toteutustapaan (liite 1).

5.2 Mielipidetiedustelun tulokset

Neljännessä kysymyksessä pyydettiin vastaajia kertomaan, mitä indie-pelejä he ovat tehneet. Vastaajista vanhimmat indie-pelit löytyivät 1990-luvun puoliväliltä, sekä jotkut olivat vielä vasta kehitysvaiheessa. Vain 3/10 vastaajista kertoi olevansa ollut useammassa pelin tekemisessä.

Viidennessä kysymyksessä kysyttiin vastaajilta sitä, mille alustalle he ovat tehneet indie-pelejä. Näistä 7/10 vastaajista oli kehittänyt pelinsä tietokonealustalle ja 7/10 mobiilialustalle. Ainoastaan 3/10 oli sillä hetkellä vain yhdelle alustalle luotuna.

Satunnaisia vastauksia:

> ”iOS, Android, Windows Phone”
> ”WP8, Android, iOS, PC, MacOSx, Web browser…siinäpä ne.”

Kuudennessa kysymyksessä pyrittiin tarkastelemaan, kuinka paljon vastaajien tekemillä peleillä on ollut suurin piirtein lataajia. Osalla peleistä ei ollut vielä lataajia, koska peliä ei ole julkaistu tai jaettu eteenpäin. Kuitenkin julkaistujen pelien lataajien määrät olivat hyvin erilaiset, sillä ne heittelivät muutamasta kappaleesta noin 10 miljoonaan. Julkaistuista peleistä keskimäärin puolet olivat hyvin menestyneitä ja toinen puoli vähemmän menestyneitä.

Seitsemännessä kysymyksessä selvitettiin, kuinka monta tekijää vastanneiden peleissä oli ollut. Näistä 2/10 oli tehnyt täysin yksin omaa peliään ja 4/10 oli tehnyt yksin tai lähes yksin kaiken pelistä. Lähes kaikki pelit oli tehty pienporukassa eli tekijöitä oli 6 tai alle. Yhdessä projektissa oli otettu enemmän pelin tekijöitä ydinporukan mukaan ja

heitä oli yhteensä 17. Lähes kaikissa tehdyissä peleissä vastaajat olivat osana pelin ohjelmoinnissa tai pelimekaniikan rakentamisessa. Noin puolet vastaajista oli ollut myös rakentamassa peleille grafiikkaa.

Satunnaisia vastauksia:

> ”Tiimin koko oli 3. Itse vastasin käyttöliittymästä, pelin ideoinnista ja äänisuunnittelusta”
> ”Tiimissä on 4 jäsentä; 1 graafikko ja 3 ohjelmoijaa. Tarvitsemme oman pelimoottorin kehitykseen enemmän ohjelmoijia, kuin graafikoita. Itse toimin pääohjelmoijana ja pelin musiikin säveltäjänä.”

Kahdeksannessa kysymyksessä haluttiin tiedustella indie-pelin ansaintamalleja. Suurin osa vastanneista oli kokeillut useampaa ansaintamallia, mutta eniten peleissä käytettiin pelin myyntiä. Mainoksia olivat useammat myös käyttäneet, mutta näiden tulokset eivät ole olleet kovinkaan positiivisia. Jotkin pelit oli luotu ilmaiseksi, mutta sisälsivät mikromaksuja.

Satunnaisia vastauksia:

> ”Android julkaistiin kokonaan ilmaisena ja Ios tarjolla oli ilmainen ja 1$ versio. Mainos budjetti oli 0 euroa eikä ihmeemmin jaksettu peliäkään netissä promota.”
> ”In-app-purchase. (=Pelin myynti)”

Yhdeksännessä kysymyksessä pyrittiin selvittämään pelin tekemisen kestoa. Vastaajista ainoastaan kaksi oli rakentanut pelin jam-tyylisesti parissa päivässä, mutta muut kestivät kahdesta kuukaudesta neljään vuoteen. Suurin osa peleistä oli tehty aikavälillä neljästä kuukaudesta kahteen vuoteen. Osa peleistä ei ollut vielä valmiita, joten vastaajat arvioivat nykyisen kulutuksen.

Satunnaisia vastauksia:

> ”Pari kk”
> ”Amiga: 4 vuotta. PC: 2 vuotta ja jatkuu edelleen.”

Kymmenennessä kysymyksessä pyydettiin mielipidettä vastaajilta liittyen indie-pelin rakentamisen kannattavuuteen. Vastaukset olivat muuten hyvin optimistisia, mutta eivät rahallisesti. Vastaajien mielestä indie-pelien tekeminen on enemmän riskipeliä kuin vakaata tulonlähdettä.

Satunnaisia vastauksia:

> ”On mutta se vaatii paljon paneutumista itse pelin tekemisen lisäksi myös pelin monetisointiin ja markkinointiin.”
> ”Pelintekemisen rahallinen kannattavuus ei ole ikinä varmaa, koska peliala on hittibisnes, kuten musiikki- tai elokuvateollisuuskin. Siihen liittyy aina riski, että vain hyvin harva edes pelaa peliä, saatika maksaa siitä.”
> ”Tähän mennessä liikevaihto on ollut noin 15 euroa puolen vuoden työstä. Ei sitä erikoisen kannattavana voi pitää. Kaupan ja teollisuuden järjestelmiä työstettäessä kassavirta olisi vastaavalla työmäärällä noin 50.000-100.000 euron hujakoilla. Niitä tässä onkin tullut koodattua neljännesvuosisata.”

Yhdennessätoista kysymyksessä syvennyttiin budjetin luomiseen sekä siihen, millä tavalla tämä kannattaisi vastaajien mielestä rakentaa. Vastaajista suurin osa suositteli budjetin rakentamista omasta pääomasta, mutta jotkut vastaajat perehtyivät jatkuviin ansaintamalleihin. Ehdotuksia olivat aiempien pelien myynnit, sijoittajat, Kelan starttiraha, Tekes:n tuet, sivutyöt kehityksen ohella, lainat ja yhteistyökumppanit.

Satunnaisia vastauksia:

> ”Säästöt / aiempien pelien myynnit. Päivätyöt ja projektityöt ovat myös ok keino hankkia rahaa mutta ne vievät yleensä sen verran paljon arvokasta työaikaa että niiden tulisi olla vasta 2. sijalla.”
> ”Siinäpä se. Kun todennäköisin tuotto-odotus on pyöreä nolla ja sovelluksia (joista suuri osa vieläpä ilmaisia) on tarjolla pelkästään AppStoressa liki miljoona, budjetin laatiminen riippuu puhtaasti käytettävissä olevasta sotakassasta. Varsinaisen pelin tekeminen maksaa sen mitä se maksaa - indie-taloissa asia hoituu yleensä energiajuoman ja menestyksestä haaveilun hinnalla - mutta kannattaa varautua siihen, että markkinointi maksaa ihan oikeaa rahaa. Ellei käy hirveä mäihä ja ohjelma "löydetään" vahingossa.”

Kahdennessatoista kysymyksessä haluttiin tietää vastaajien indie-pelisuosikeista. Kysymyksen ideana oli tiedustella sitä, onko jotain tiettyä tyyppiä, mitkä olisivat enemmän suosiossa. Valitettavasti jokainen vastaus oli erilainen toisistaan. Yksi asia kuitenkin oli useassa vastauksessa yhteinen, ja se oli, että suosikki-indie-pelit olivat tunnettuja innovaatioista.

Satunnaisia vastauksia:

> ”PC-puolella tsekkiläinen Bohemia Interactive työsti aikanaan verrattomia Flash-Point-sotasimulaatioita, joista rahakkaat jenkkitalot ovat selvästi katsoneet mallia omissa Battlefieldseissään ja Medal of Honoreissaan.”
> ”Double Finen uusin peliprojekti Broken Age on kenties mielenkiintoisin. Sen Kickstarter-menestys oli valtava ja innoitti monia muita pelejä hakemaan joukkorahoitusta.”

5.3 Mielipidetiedustelun analysointi

Mielipidetiedustelu oli skaalaltaan melko pieni, mutta vastauksista saatiin hyvin pohjatietoa indie-pelin kehityksestä. Lähes kaikki olivat vastanneet kaikkiin kysymyksiin ja vastaukset olivat asiallisesti kirjoitettuja. Ainoastaan 3/10 halusi pitää identiteettinsä anonyyminä.

Vastaajat suosivat tehdä pelejä useammalle alustalle, jotka olivat kaikki mobiili- tai tietokonekäyttöjärjestelmiä. Latauksissa erot olivat suuria tai peliä ei ollut vielä julkaistu. Pelien tekijämäärät olivat enimmäkseen 1 - 6 henkilöä, ja suurin osa vastaajista oli mukana pelin ohjelmoinnissa. Peleistä tehtiin voittoa myymällä peliä, mikromaksuilla ja mainoksilla, joista mainokset olivat tuottaneet huonoiten. Pelin kehityksen kesto vaihteli kahdesta päivästä neljään vuoteen, mutta suurin osa peleistä luotiin neljästä kuukaudesta kahteen vuoteen.

Mielipidekysymyksissä vastaukset olivat suurimmaksi osaksi positiivisia, ja monesti pyrittiin painottamaan perusteellista työtä. Vastaajien mielestä indie-pelin tekeminen vaatii työtä, mutta varsinainen menestys on riskipeliä. Budjetin luomisessa vastaajat suosivat useampaa tyyliä, mutta suurin osa suositteli rahan keräämistä ennakkoon tai sivutyön hankkimista pelin kehityksen ohelle. Viimeisessä mielipidekysymyksessä ei ollut lainkaan samoja vastauksia, mutta useampi vastaus viittasi peleihin, jotka ovat kuuluisia innovaatiosta.

6 Yhteenveto

Opinnäytetyön alussa tavoitteena oli tutkia indie-peliohjelmointia sekä luoda oma peli valmiiksi asti. Pelin luonti jäi hieman kesken, sillä luontiprosessi oli odotettua pidempi, vaikka pelin mekaniikka toimiikin jo täysin. Työn edetessä pelin rakentamisen vaatimukset laajenivat, mikä johtui liian vähäisestä taustatutkimuksesta ennen opinnäytetyön aloittamista. Opinnäytetyö pitää sisällään kompaktin paketin työssä käydyistä aiheista: indiestä peliohjelmoijan näkökulmasta, Android-alustasta, Android-sovelluksien kehittämisestä ja siitä, kuinka tehdään yksinkertainen 2D-pulmanratkaisupeli.

Opinnäytetyössä käy ilmi, että jokaisen, joka haluaa indie-peliohjelmoijaksi, tulisi selvittää enemmän alasta kuin media antaa ilmi. Indie alana ei ole kovinkaan kannattavaa

aloittelijoille, ja lisäksi se vaatii hyvin laajan osaamisen pelkän pelin kehittämisen lisäksi. Indie-pelinohjelmointi ei ole pelkästään pelin rakentamista, vaan tekijän täytyy laajentaa osaamista ansaintamalleihin, käyttäjien vaatimaan lopputulokseen, testaamisen tärkeyteen sekä markkinointiin ja promotointiin.

Android alustana on erittäin hyvä valinta, mutta opinnäytetyössä käy ilmi, ettei välttämättä kannata turvautua vain yhteen alustaan. Yhteen alustaan turvautuminen on yleensä nopea ratkaisu ja toimii niin sanotusti jään rikkojana, mutta tuotto saattaa jäädä vähemmälle. Toinen asia, joka kävi ilmi Androidista, liittyi ohjelmistorajapintarajoituksiin. Androidilla päivityksien myötä tulee uusia ominaisuuksia ja näiden kanssa pitää olla tarkkana, mikäli haluaa kehittää ohjelmansa vanhemmille versioille.

Ulkopuolisille indie-pelin rakentaminen saattaa tuntua hankalalta ja pitkäaikaiselta puuhalta. Kyseistä peliä on tehty opinnäytetyön ohella, joista pelin luontiin on kulunut noin kuukauden verran. Pelin kehitysvaiheet käytiin tarkasti läpi ja pyrittiin luomaan sillä tavalla, että ohjelmoinnista tietämätön voisi myös ymmärtää lähdekoodin pätkiä. Opinnäytetyössä ei ole kerrottu opas tyylisesti pelin rakentamista vaan käyty läpi tärkeitä elementtejä, jotka ovat osana peliä.

Mielipidetiedustelu on tärkeä osa opinnäytetyötä, josta saa selville jo aikaisemmin alalla työskenteleviltä mietteitä alasta. Tiedustelussa on referoitu ja analysoitu vastaukset.

Lähteet

1 Jaakko Suominen, Pelitutkimuksen vuosikirja 2009 s. 49–56, Kolme näkökulmaa independent-peleihin [e-kirja] <http://www.pelitutkimus.fi/wp-content/uploads/2009/08/ptvk2009-04.pdf> Viitattu: 19.11.2013.

2 Global Game Jam, FAQ [verkkodokumentti] <http://globalgamejam.org/faq> Viitattu: 11.11.2013.

3 Stackoverflow, difference between project management and process management [closed], [verkkodokumentti] <http://stackoverflow.com/questions/604748/difference-between-project-management-and-process-management> Viitattu: 11.11.2013.

4 Wikipedia, List of game engines [verkkodokumentti] <http://en.wikipedia.org/wiki/List_of_game_engines> Viitattu: 11.11.2013 .

5 Wikipedia, Game Development [verkkodokumentti] <http://en.wikipedia.org/wiki/Game_development> Viitattu 11.11.2013.

6 Tech Crunch, How Free Apps Can Make More Money Than Paid Apps [verkkodokumentti] <http://techcrunch.com/2012/08/26/how-free-apps-can-make-more-money-than-paid-apps/> Viitattu: 11.11.2013.

7 My Android News, The Problem Of Micropayments In Android Games, And How To Avoid [verkkodokumentti] <http://myandroidnews.com/the-problem-of-micropayments-in-android-games-and-how-to-avoid/> Viitattu 11.11.2013.

8 Wikipedia, Joukkorahoitus [verkkodokumentti] <http://fi.wikipedia.org/wiki/Joukkorahoitus> Viitattu 11.11.2013.

9 MOL, Starttirahalla yrittäjäksi, [pdf-verkkodokumentti] <http://www.mol.fi/mol/fi/99_pdf/fi/01_esitteet/starttiraha.pdf> Viitattu 11.11.2013.

10 Tekes, Slide Share, Milloin Skenestä rahoitusta pelinkehitykseen? [kalvo-verkkodokumentti] <http://www.slideshare.net/TekesICT/skene-rahoitus> Viitattu 11.11.2013.

11 Wikipedia, Android software development [verkkodokumentti] <http://en.wikipedia.org/wiki/Android_software_development> Viitattu: 14.11.2013.

12 VisualGBD, Using Visual Studio to Develop Native Android Code [verkkodokumentti] <http://visualgdb.com/tutorials/android/> Viitattu: 14.11.2013.

13 Android Developers, Building and Running from the Command Line [verkkodokumentti] <http://developer.android.com/tools/building/building-cmdline.html> Viitattu 14.11.2013.

14 Android Developers, Getting Started with Android Studio [verkkodokumentti] <http://developer.android.com/sdk/installing/studio.html> Viitattu: 14.11.2013.

15 Android Developers, Get the Android SDK [verkkodokumentti] <http://developer.android.com/sdk/index.html> Viitattu: 14.11.2013.

16 Android Developers, Android Projects [verkkodokumentti] <http://developer.android.com/tools/projects/index.html> Viitattu: 14.11.2013.

17 Wikipedia, Java (programming language) [verkkodokumentti] <http://en.wikipedia.org/wiki/Java_(programming_language)> Viitattu: 14.11.2013.

18 Wikipedia, XML [verkkodokumentti] <http://en.wikipedia.org/wiki/XML> Viitattu: 14.11.2013.

19 Android Developers, android.animation [verkkodokumentti] <http://developer.android.com/reference/android/animation/package-summary.html> Viitattu: 14.11.2013.

20 Android Developers, Platform Versions [verkkodokumentti] <http://developer.android.com/about/dashboards/index.html> Viitattu 30.1.2014.

21 Wikipedia, Google Play [verkkodokumentti] <http://en.wikipedia.org/wiki/Google_Play> Viitattu 20.11.2013.

22 Google Play, Accept Developer Agreement [verkkodokumentti] <https://play.google.com/apps/publish/signup/> Viitattu 15.11.2013.

23 AndroLib, Distribution of downloads in the Android Market [kuva] <http://www.androlib.com/gd/stats/downloadrepartition.aspx> Viitattu 15.11.2013.

24 AndroLib, Distribution of free and paid apps in Android Market [kuva] <http://www.androlib.com/gd/stats/freepaidapp.aspx> Viitattu 15.11.2013.

25 AndroLib, Distribution of Apps and Games in Android Market [kuva] <http://www.androlib.com/gd/stats/cattype.aspx> Viitattu 15.11.2013.

26 Wikipedia, List of mobile software distribution platforms [verkkodokumentti] <http://en.wikipedia.org/wiki/List_of_mobile_software_distribution_platforms> Viitattu 15.11.2013.

27 Business Insider, Smartphone Market Share By OS (GLOBAL) [kuva] <http://static5.businessinsider.com/image/50a3b2546bb3f72b7b000007-960/smartphone-market-share.png> Viitattu: 17.11.2013.

28 IDC, Press Release [verkkodokumentti] <http://www.idc.com/getdoc.jsp?containerId=prUS24442013> Viitattu 17.11.2013.

29 Betanews, Tablet market share Trends -- Q3 2013: branded Android vs iPad [verkkodokumentti] <http://betanews.com/2013/11/04/192717/> Viitattu 17.11.2013.

30 Java For Each, Creating An Android Game in a Day [verkkodokumentti] <http://www.jforeach.com/creating-an-android-game-in-a-day/80> Viitattu 18.11.2013.

31 Wikipedia, Arkusfunktiot [verkkodokumentti] <http://fi.wikipedia.org/wiki/Arkusfunktiot> Viitattu 18.11.2013.

32 Android Developers, Math [verkkodokumentti] <http://developer.android.com/reference/java/lang/Math.html> Viitattu 18.11.2013.

33 Linux.com, Android Application Development Tutorial: How to Handle Multi-Touch [verkkodokumentti] <http://www.linux.com/learn/tutorials/715293-android-application-development-tutorial-how-to-handle-multi-touch> Viitattu 18.11.2013.

34 Stackoverflow, Android Playing 1 sound at a time [verkkodokumentti] <http://stackoverflow.com/questions/6886495/android-playing-1-sound-at-a-time> Viitattu 18.11.2013.

35 Wikipedia, Bug tracking system [verkkodokumentti] <http://en.wikipedia.org/wiki/Bug_tracking_system> Viitattu 19.11.2013.

Liite 1: Mielipidetiedustelu

Indie-pelin tekijöiden haastattelu

(Huom. Mikäli pelejä on useampia niin kaikkia ei tarvitse ottaa mukaan)

1. **Nimi:** *
2. **Oletko tällä hetkellä Indie-pelin tekijä ja missä? (yritys, toiminimi jne.)**
3. **Jos et ole tällä hetkellä, oletko ollut mukana Indie-pelin teossa ja missä? (yritys, toiminimi jne.)**
4. **Mitä Indie-pelejä olet tehnyt?** *
5. **Mille alustalle olet tehnyt Indie-pelejä?** *
6. **Kuinka paljon pelillä/peleillä on ollut lataajia? (suurin piirtein)** *
7. **Kuinka monta tekijää pelissä/peleissä oli ja mikä oli sinun työsi?** *
8. **Millä tavalla haitte tuottoa Indie-pelillä/peleillä? (esim. pelin myynti, mainokset tai mikromaksut)**
9. **Kuinka kauan kesti tehdä peli(t)?**
10. **Mielipiteesi onko Indie-pelin tekeminen kannattavaa?**
11. **Millä tavalla mielestäsi kannattaa rakentaa Indie-pelin tekemisen budjetti?**
12. **Mikä on mielenkiintoisin Indie-peli mielestäsi?**
13. **Lisää sähköpostisi tähän, jos haluat ilmoituksen, että tulen käyttämään vastauksiasi opinnäytetyössäni.**
14. **Mikäli haluat vastata anonyymisenä, laita rasti ruutuun.**
    (vastauksia ei tulla julkaisemaan, mutta saatetaan käyttää referoinnissa)

Liite 2: Buglog.txt

* = Korjattu, ! = Kriittinen, ? = Hämärä, -> = Korjaustapa tiedossa, / = Kesken, + = ja

/Animaatio

*Painovoima toimii epämääräisesti

*Piirto tapahtuu päällekäin

*Äänet lopettaa toiminnan jos liikaa spämmii näyttöön

*Liikkumisen kosketus ei toimi halutulla akselilla

! + ->Peli kaatuu muistin puutteesta joillakin kännyköillä

->Maaliin päästyä takaisin-nappi tuo eteen välivalikon

*Pelin restart ei toimi normaalisti