Jesse Kotiranta

ANDROID SOVELLUKSEN KEHITTÄMINEN

Tietojenkäsittelyn koulutusohjelma

2015

samk

Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

ANDROID SOVELLUKSEN KEHITTÄMINEN

Kotiranta, Jesse
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Toukokuu 2015
Ohjaaja: Grönholm, Jukka
Sivumäärä: 26
Liitteitä:

Asiasanat: Android, Sovelluskehitys, Projektinhallinta

Opinnäytetyön aiheena oli Android sovelluksen kehittäminen projektinhallinnan näkökulmasta. Opinnäytetyössä käydään lävitse kehitysvälineitä ja menetelmiä liittyen sovellustuotantoon. Työssä seurataan myös Möllykkä-nimisen pelin kehitystä.

DEVELOPING ANDROID SOFTWARE

Kotiranta, Jesse
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Business Information Systems
May 2015
Supervisor: Grönholm, Jukka
Number of pages: 26
Appendices:

Keywords: Android, Software development, Project Management

The purpose of this thesis was to view development of android software through project management. In this work I will go through few IDE:s and development methods. I will also tell about the development process of mobile game project.

SISÄLLYS

1 JOHDANTO

Opinnäytetyöni käsittelee sovelluskehitystä Androidille. Opinnäytetyössä tullaan perehtymään kysymyksiin; mikä on Android, miksi valita Android muiden alustojen sijaan sekä sovellustuotannon projektinhallinnallisiin kysymyksiin.

Työssä seurataan samalla Möllykkä-nimisen mobiilipelin kehitykseen liittyvää projektia. Mobiilipelit on alue jossa suomi on menestynyt mm. Rovion ja Supercellin kaltaisten yritysten toimesta. Työssä tullaan käymään näiden yritysten nousutarinat lävitse sekä mobiilipelien nousua muutenkin. Mobiilipeleissä on myös useita erityyppisiä peli genrejä joista ”Kasuaali”-on tällä hyvin suosittua. Tulen vertailemaan mobiilipeligenrejen suosiota muihin alustoihin nähden.

Ohjelmistokehitys lähtee useimmiten liikkeelle hyvästä ideasta. Juuri hyvän idean löytäminen on myös haastavin sovelluskehityksen osa-alue, mutta en keskity tässä työssä varsinaiseen innovointiin, vaan valmiin idean tarkempaan määrittelyyn sekä myöhemmin vaihtoehtoisiin toteutustapoihin. Android-sovellusta kehittäessä ohjelmointikielenä on Java sekä tietokantana toimii SQLite.

Sovelluskehitystyö on alustasta riippumatta varsin samanlaista, tehtiinpä sovellusta sitten PC:lle tai mobiilikäyttöön. Suunnitteluprosessi on työn onnistumiselle kaikista merkitsevin. Sanotaan että hyvin suunniteltu on puoliksi tehty, mutta sovelluskehitysprojekteissa koodausta on yleensä vain 20–30 % työnmäärästä ja loppuosuus pääosin suunnittelua.

Alustana Android on hyvin houkutteleva sen avoimuuden vuoksi. Sovelluskehittäjät eivät joudu maksamaan lisenssimaksuja elleivät halua julkaista sovellustansa Play Storessa, silloinkin maksu on tosin pieni ja kertaluontoinen.

2 ANDROID

Android-käyttöjärjestelmän menestys lähti liikkeelle vuonna 2005 tapahtuneella yrityskaupalla, jossa Google osti Android Inc. yrityksen ja päätti määrätietoisesti ja isolla rahalla lähteä kehittämään mobiililaitteille tarkoitettua käyttöjärjestelmää. Android on siis Linuxiin pohjautuva mobiili alustoille tarkoitettu käyttöjärjestelmä ja se on avoimen lähdekoodin tuote, jota useat valmistajat käyttävät puhelimissaan.

2.1 Miksi Android

Googlen mukaan Android on asennettuna tällä hetkellä sadoille miljoonille mobiililaitteille sekä joka päivä yli miljoona uutta Android laitetta aktivoidaan. Android käyttäjät lataavat myös kuukausittain yhteensä 1,5 biljoonaa sovellusta ja peliä Google Play Store. (LÄHDE http://developer.android.com/about/index.html 18.1.2015)

Tämä tilasto puhuu jo puolestaan sen osalta, että Android on sellainen alusta, jolle kannattaa kehittää sovelluksia, mikäli haluaa niiden menestyvän. Sovelluksen kehittäminen Androidille ei yksin takaa sille menestystä, vaan menestykseen vaikuttaa myös markkinointi, sovelluksen laatu ja erityisesti sovelluksen idea sekä toteutus.

Google Play Store on myös todella hyvä syy tehdä sovellus juuri Androidille. Google Play Storessa pääset helposti jakamaan sovelluksesi jokaisen Android käyttäjän saataville. Google Play Storeen kuka vain voi viedä oman sovelluksensa. Kehittäjätilin rekisteröintimaksu on ainoastaan 25 dollaria. Android on mobiililaitteiden lisäksi käytössä myös kodinkoneissa, älykelloissa, autoissa, tableteissa, televisioissa ja monissa muissa laitteissa. Sen puolesta androidille voidaan suunnitella ja toteuttaa todella monenlaisia ja monipuolisia sovelluksia.

Androidissa on toki myös omat haasteensa. Alustan suosion takia se houkuttelee myös rikollisia tekemään haittaohjelmia Androidille. Koska Google Play Storeen saa käytännössä kuka vain julkaista sovelluksia jakoon niin loppukäyttäjän on tärkeää olla hyvin tietoinen sovellusten käyttöoikeuksista.

2.2 Kehitysvälineet

Koska projekti tullaan toteuttamaan Androidille, niin kielenä tulee myös toimimaan Java jonka myötä myös kehitysvälineet tulee valita sen mukaan. Java on oliopohjainen ohjelmointi kieli ja sitä kehittää Sun Microsystems. Java on lisensoitu GNU GPL lisenssin alaiseksi mikä tarkoittaa käytännössä sitä että Javaa saa käyttää aivan miten itse haluaa. Javaa voi ohjelmoida käyttäen mitä editoria vain haluaa. Pelkän Notepad:in käyttö ei ole silti tehokkain valinta varsinkin kun on olemassa monia ilmaisia työkaluja. Tunnetuin kehitysväline Javalle lienee Eclipse. Eclipse julkaistiin vuonna 2001 ja sitä kehittää tänäkin päivänä Eclipse yhteisö. Emme silti valinneet Eclipseä työkaluksemme.

Vaihtoehtoja on monia mutta projektissa päädyimme toteuttamaan PC:llä pyörivän demo version pelistä käyttäen kehitys ympäristönä IntelliJ IDEA:a. IntelliJ IDEA:n käyttämiseen vaikutti sen lähiaikoina saamat kehut parhaimpana ohjelmointityökaluna Javalla koodaamiseen. Sovellus oli todella helppokäyttöinen ja uskoisin sen käytön nopeuttaneen demon valmistumista. Vaikka valitsimme kyseisen IDE:n projektiin, niin valinta on loppujenlopuksi aina täysin subjektiivinen.
(LÄHDE: http://www.infoworld.com/article/2683534/development-environments/infoworld-review--top-java-programming-tools.html 7.5.2015)

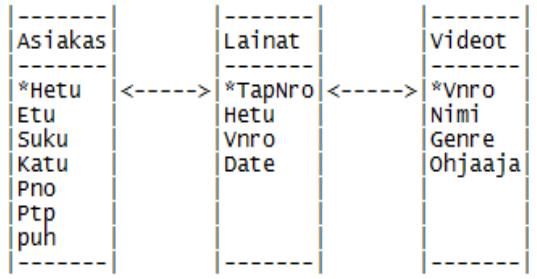
Itse projektin koodaaminen päätettiin toteuttaa käyttäen Android Studiota. Android studiolla voidaan emuloida eri Android versioita ja näin ollen testata sovelluksen toimintaa hyvin tarkasti. Tämä ominaisuus oli hyvin tärkeä meille työkalun valinnan ollessa esillä.

Halusimme käyttää myös tietokantaa sovelluksessamme. Androidilla on ainoastaan yksi vaihtoehto tähän käyttöön ja se on SQLite. SQLiten käytön etuna on se miten se on ainoastaan n. 225 kt:n kokoinen C-kielellä toteutettu kirjasto. SQLite eroaa monista muista tietokannoista siten että siinä tyypitys ei ole sarakekohtainen vaan arvokohtainen. Tämä johtaa siihen että tietokannan ei ole pakko noudattaa ACID-mallia.

Tietokannan tulisi noudattaa silti ACID (Atomicity, Consistency, Isolation, Durability) mallia jonka tarkoitus olisi taata tietokannan eheys aina. (Jönkkäri, 2007, 36)

SQLitestä puuttuu myös viiteavainten pakollisuus, se ei tarkoita että niitä ei kannattaisi käyttää tietokantaa suunnitellessa.

Tietokannan suunnittelu alkaa siitä kun pohditaan mitä tietoa on tarve tallentaa ohjelman kannalta. Esimerkiksi jos ajatellaan videovuokraamon toimintaa, niin tietokantaan tarvitaan tiedot siitä mitä videoita löytyy, ketkä ovat asiakkaita ja mitä asiakkaat ovat lainanneet. Tämä on hyvin yksinkertainen esimerkki tietokannasta joka koostuu kolmesta taulusta.



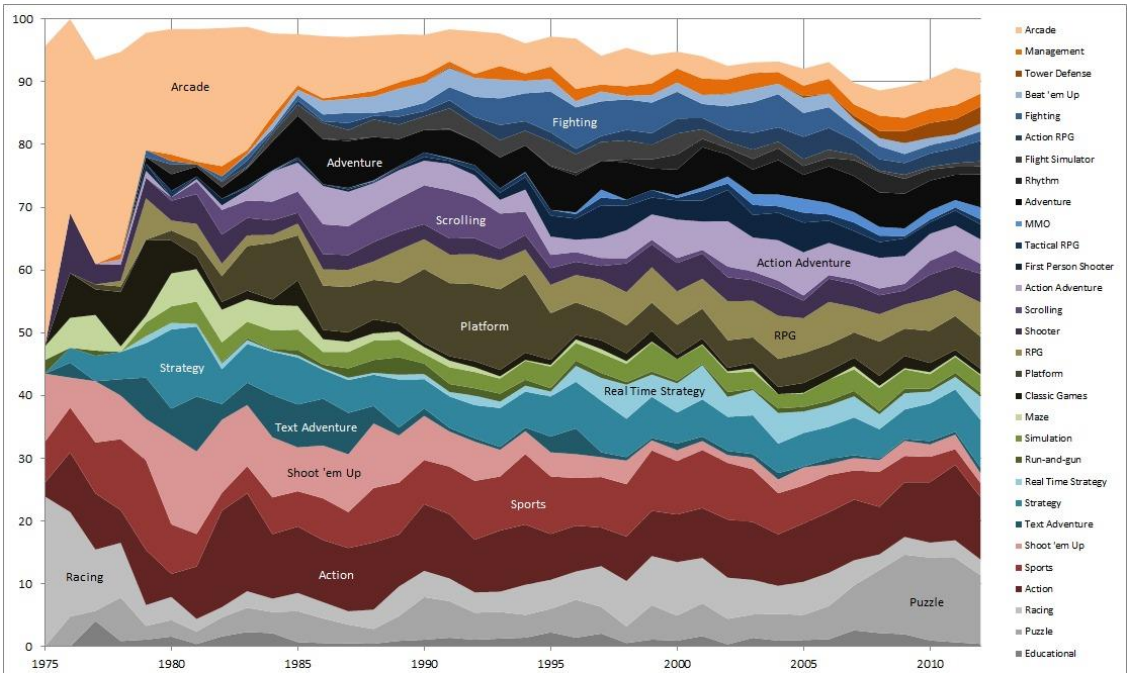
(Lähde: http://fi.wikipedia.org/wiki/ACID 7.5.2015)
(Lähde: http://fi.wikipedia.org/wiki/SQLite 7.5.2015)

3 MOBIILIPELIGENRET

Lähivuosien aikana moni suomalainen yritys on tehnyt nimeä nimenomaan pelejä tekemällä. Tunnetuimpia yrityksiä lienee Rovio Angrybirds sarjansa kanssa sekä SuperCell Clash of Clans sarjansa kanssa. Tietenkään se että toteutat mobiilipelin ja pistät sen myyntiin, ei tee sinusta saman tien menestyjää. Se vaatii myös taitavaa markkinointi osaamista sekä laadukkaan tuotteen mistä ihmiset haluavat maksaa.

Aiemmat firmat ovat itse julkaisseet oman pelinsä, vaihtoehtoisesti voidaan käyttää myös eri julkaisijaa. Suomalaisittain varmasti tunnetuin esimerkki on RedLynx joka kehittää Trials-pelisarjaa käyttäen ubisoftia julkaisijana. Julkaisijaa käyttäessä etuna on se että ei tarvitse itse tehdä mitään markkinointiin tai jakamiseen liittyvää ja voit käyttää enemmän aikaa itse tuotteen kehityksessä. Julkaisijalla on silti hyvin paljon sananvaltaa siitä, mihin suuntaan sovellusta kehitetään.

Peleihin liittyy paljon erilaisia genrejä. Se että mikä on suosituin peli genre minäkin hetkenä on hankala mitata mutta sen sijaan voidaan muodostaa tilastoa siitä minkä peligenren pelejä minäkin vuonna on julkaistu ja minkä verran.

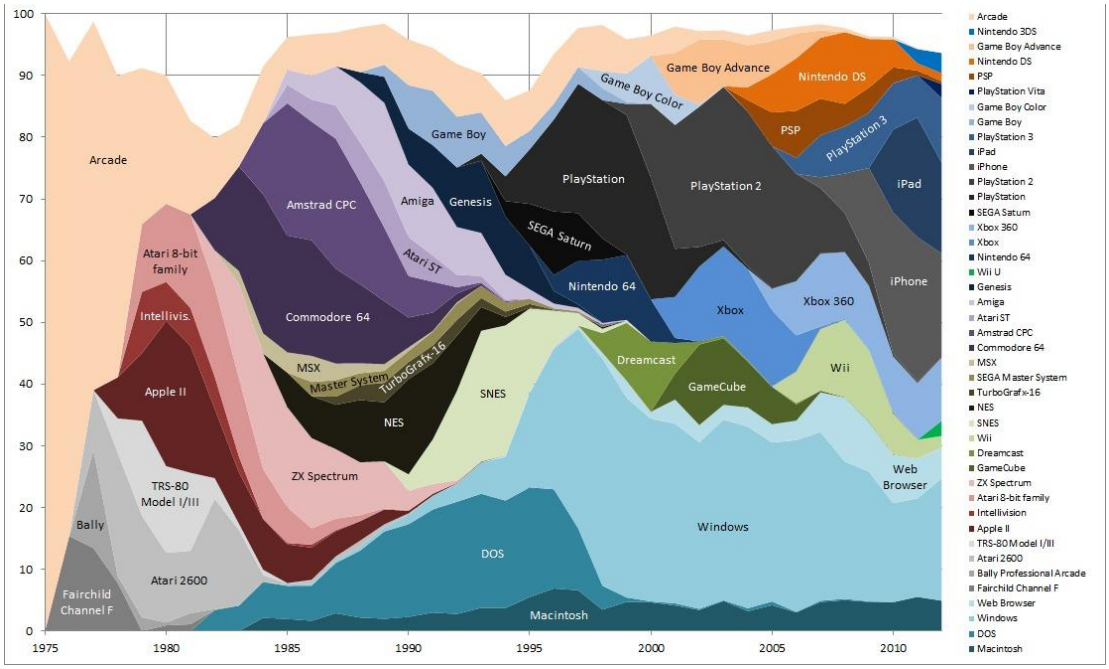


(Viitattu http://www.gameinformer.com/b/features/archive/2012/12/27/tracking-the-popularity-of-videogame-genre.aspx 29.4.2015)

Yllä olevasta tilastosta voimme havaita miten 75-luvun alussa Arcade pelit muodostivat puolet siitä määrästä pelejä mitä silloin julkaistiin. Sittemmin on alkanut syntyä erilaisia peligenrejä jotka ovat ajan mittaan kasvattaneet suosiotaan.

Tällä hetkellä Puzzle pelien suosiota selittänee paljolti mobiililaitteiden suosio ja pienten pelien pelaamisen tarpeen kuten Sudoku. Pelin genre ei määritä pelin suosiota mutta se vaikuttaa siihen myös. Ihmiset haluavat pelata tiettyä genreä tietyllä laitteella. Esimerkiksi Sudokun konsoliversio tuskin saavuttaisi yhtä valtavaa menestystä kuin vastaava peli mobiilipelinä.

Genrejen suosioon vaikuttaa siis myös alusta, seuraavalla sivulla on tilastoa pelien määrästä tiettyä alustaa kohti.



(Viitattu [http://www.gameinformer.com/b/features/archive/2012/12/27/tracking-the-popularity-of-videogame-genre.aspx 29.4.2015](http://www.gameinformer.com/b/features/archive/2012/12/27/tracking-the-popularity-of-videogame-genre.aspx))

Voimme tulkita tilastosta sen miten uuden alustan tullessa markkinoille siihen tulee alussa valtava julkaisu buumi. joka alkaa vähitellen laskea seuraavan sukupolven laitteen julkaisun lähentyessä.

Ainoa tasaisen varma alusta, millä on ollut pelaajia jo usean vuoden ajan, on Windows. Vaikka pc-pelaamisen kuolemasta uutisoidaan vähintäänkin vuosittain, niin käytännössä se on edelleen alustana sellainen mille tehdään tilastollisesti eniten pelejä. Iso kohderyhmä tekee kehittäjille myös lisää paineita. Pelin on oltava kyllin erilainen jotta se erottuu massasta.

Pelien ei ole pakko noudattaa tietyn peli genren raameja vaan niitä voidaan myös yhdistellä oman luovuuden mukaan. Esimerkkinä Möllykkä pelimme yrittää yhdistää casuaali genreen RPG-pelien (Role Playing Game) elementtejä.

4 PROJEKTINHALLINTAMENETELMÄT

Sovellustuotannossa käytetään monenlaisia projektinhallinta menetelmiä. Näistä varmasti tunnetuimmat ovat vesiputous malli sekä ketterä kehitys. Projektinhallinta menetelmän käyttämisen hyödyn huomaa varsinkin silloin kun sellaista ei käytetä. Mikäli projektipäällikkö puuttuu, niin hyvin usein asioista päädytään keskustelemaan loputtomiin ilman että saadaan nopea päätös ja itse työ voisi jatkua.

4.1 Suunnittelun määrittely

Sovelluskehityksen suunnittelu tulee määritellä. Suunnitteluun on apuna paljon erilaisia kaaviotyyppejä, kaikkia niitä ei ole järkevää tehdä sovelluksen suunnittelun aikana, sen sijaan olisi tärkeää tehdä ainoastaan ne mille todellisuudessa löytyy käyttöä sovelluksen kehityksen kannalta. Mieltäisin tärkeimmäksi asiaksi sovellussuunnittelussa tietokantasuunnittelun, sillä sitä on hyvin ikävä alkaa jälkeenpäin muokkaamaan kun kaikki tiedot ovat sisällä ja sovellusta on alettu kehittämään sen ympärille.

Suunnitellessa tulevaa on myös hyvä tehdä päätös siitä millä menetelmällä ohjelmistoa tullaan kehittämään eteenpäin. Vaihtoehtoja on monia ja nykyään suosiota niittää ketterät kehitysmallit, joissa pyritään pitämään säännöllisiä useita palavereita tiimin kanssa joista edetään sprinttiin jotta kehitys pysyisi nopeatempoisena.

Ohjelmistokehitystä varten on yleensä versionhallintatyökalu käytössä. Versionhallinta helpottaa nimensä mukaisesti eri ohjelmisto versioiden hallintaa. Versionhallintaa käytettäessä on helppo palata katsomaan mikä on muuttunut aiempaan nähden. Vaihtoehtoisia palveluita on todella paljon mutta maininnan arvoinen esimerkki voisi olla Linus Torvaldin kehittämä Git (mm. GitHubissa on nykyisin 3,4 miljoonaa käyttäjää sekä 16,7 miljoonaa repositoryä).

4.2 Kehitysmenetelmät

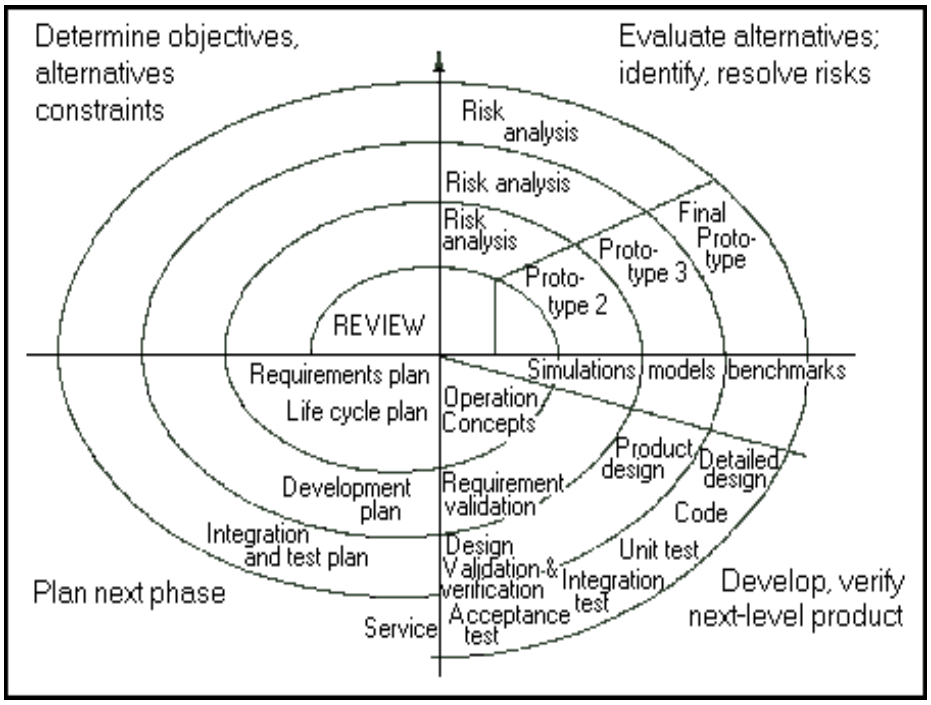
Kehitysmenetelmiä on paljon, löytyy melko uusia ketteriä kehitys menetelmiä sekä myös perinteisempiä vanhempia kehitysmenetelmiä jotka ovat edelleen käytössä esimerkkinä vesiputous- ja prototyyppi mallit. Mikään menetelmä ei ole välttämättä parempi suhteessa toiseen, kyseessä on ennemminkin tapa miten haluat kehittää ohjelmistoa. Päätös tietyn kehitysmenetelmän käytöstä tulee tehdä aina tilannekohtaisesti kaikki muuttujat huomioon ottaen.

4.2.1 Scrum

Scrum on ketterä kehitysmenetelmä joka kehitettiin japanissa jo vuonna 1986 mutta tuli yleisesti käyttöön vasta vuonna 2001. Tuona vuonna tehtiin Agile Alliance niminen järjestö joka on alkanut puhua ketterien kehitysmenetelmien puolesta. Nykyisin Scrum on yhtenä ketteränä kehitysmenetelmänä yleisesti käytössä ohjelmistokehityksessä. Scrumissa toteutetaan 1-4 viikon sprinttejä. Jokaisen sprintin alussa on palaveri, missä päätetään mitä kehitysjonosta toteutetaan. Sprintin jälkeen pidetään uusi palaveri aiemmasta sprintistä jolloin katsotaan asioiden joko onnistuneen tai epäonnistuneen. Seuraavan sprintin aihe päätetään vasta uuden sprintin alkupalaverissa.

4.2.2 Prototyyppimalli

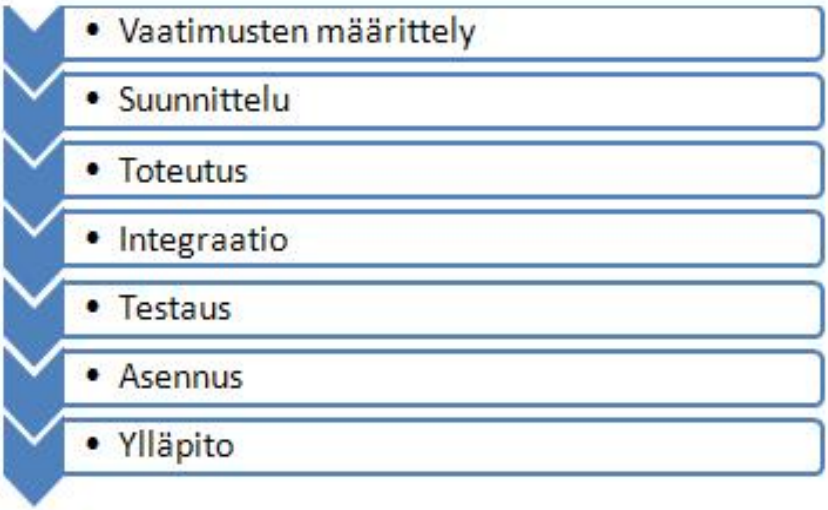
Prototyyppimallissa ohjelmistoa kehitetään spiraalimaisesti eli alussa luodaan luonnos käyttöliittymästä jonka jälkeen se hyväksytetään asiakkaalla, sitten luodaan toiminnallisuutta joka taas hyväksytetään asiakkaalla, tätä jatketaan kunnes spiraalin ensimmäinen kierros on menty täysin lävitse. Spriraalin ensimmäisen kierroksen jälkeen aletaan syventämään aiempia osioita aina aloittaen siitä kohtaa mistä spiraali on alkanut aina enemmän ja enemmän niiltä osin mihin asiakas haluaa muutoksia. Spiraalimalli on todella käytännöllinen silloin kun ei ole täysin tiedossa mitä asiakas haluaa tai asiakkaan tarpeet muuttuvat kesken projektin. Alla havainnollistava kuva spiraalimallin mahdollisista vaiheista. Tässä kuvassa spiraalin neljäs osuus on tuotteen hyväksyttämistä asiakkaalla.



(Kuva viitattu http://www.ibm.com/developerworks/rational/library/dec04/pollice/pollice-fig2.gif 23.3.2014)

4.2.3 Vesiputousmalli

Vesiputousmallissa sen sijaan edetään vaiheesta vaiheeseen palaamatta aiempaan vaiheeseen, ihanteellisessa vesiputousmallin mukaisen projektin alussa suunnitellaan koko tuleva ohjelma täydellisesti, jonka jälkeen siirrytään toteuttamaan ohjelmistoa. Tosin tämä ei lähestulkoonkaan aina toteudu juuri näin, johtuen siitä miten asiakas haluaa kesken tuotannon muuttaa ohjelmiston vaatimuksia sekä muista mahdollisista syistä johtuen. Muut mallit useimmiten pohjaavat jonkin verran vesiputousmalliin. Alla kuva vesiputousmallin vaiheista.



(Kuva viitattu http://fi.wikipedia.org/wiki/Vesiputousmalli 23.3.2014)

4.3 Testaus

Ohjelman testausta voisi luonnehtia lähes tärkeämmäksi kuin ohjelman luomista. Ohjelmalla mikä ei toimi ei tee yhtään mitään. Yleensä on suotavaa että ohjelmiston testaa eri tiimi kuin sen on luonut sillä löydetyt virheet tarkoittavat lisätöitä koodaajille. Ohjelmiston testaamiseen voidaan luoda toinen ohjelma joka suorittaa pää ohjelmaa useita kertoja virheitä etsien.

Sovellusta testataan usein samalla kun sitä koodataan. Android sovelluksia voi testata mm. emulaattoria hyödyntäen. Tämän ominaisuuden tarjoaa ainakin Android Studio. Emulaattorin käyttämisen etuna oikeaan on kustannusten säästäminen. Näin ollen voi vähävaraisempikin kehittäjä testata sovelluksiaan ilman että joutuu hankkimaan kalliita laitteita sitä varten.

Todennäköisin virhe ainakin amatööri tasolla on sellainen mikä kaataa sovelluksen. Ne on onneksi helppo löytää kun käynnistää sovelluksen ja menee manuaalisesti sen toiminnot lävitse. Mitä laajempi sovellus on, niin sitä haastavampaa sovelluksen testaaminen on. Ei voida koskaan olettaa että loppukäyttäjät käyttäisivät sovellusta juuri siten miten kehittäjä on tarkoittanut. Siksi jo testausvaiheessa on hyvä yrittää suorittaa toimintoja vastoin niiden toiminta tarkoitusta.

4.4 Elinkaari

Ohjelmiston elinkaaren ei pitäisi päättyä siihen kun sovellus on tehty määritysten mukaan ja projekti asiakkaan kanssa on ohitse. Sen sijaan koodarin on hyvä pidentää ohjelmiston elinkaarta tekemällä sopimus asiakkaan kanssa myös ohjelman tulevasta ylläpidosta.

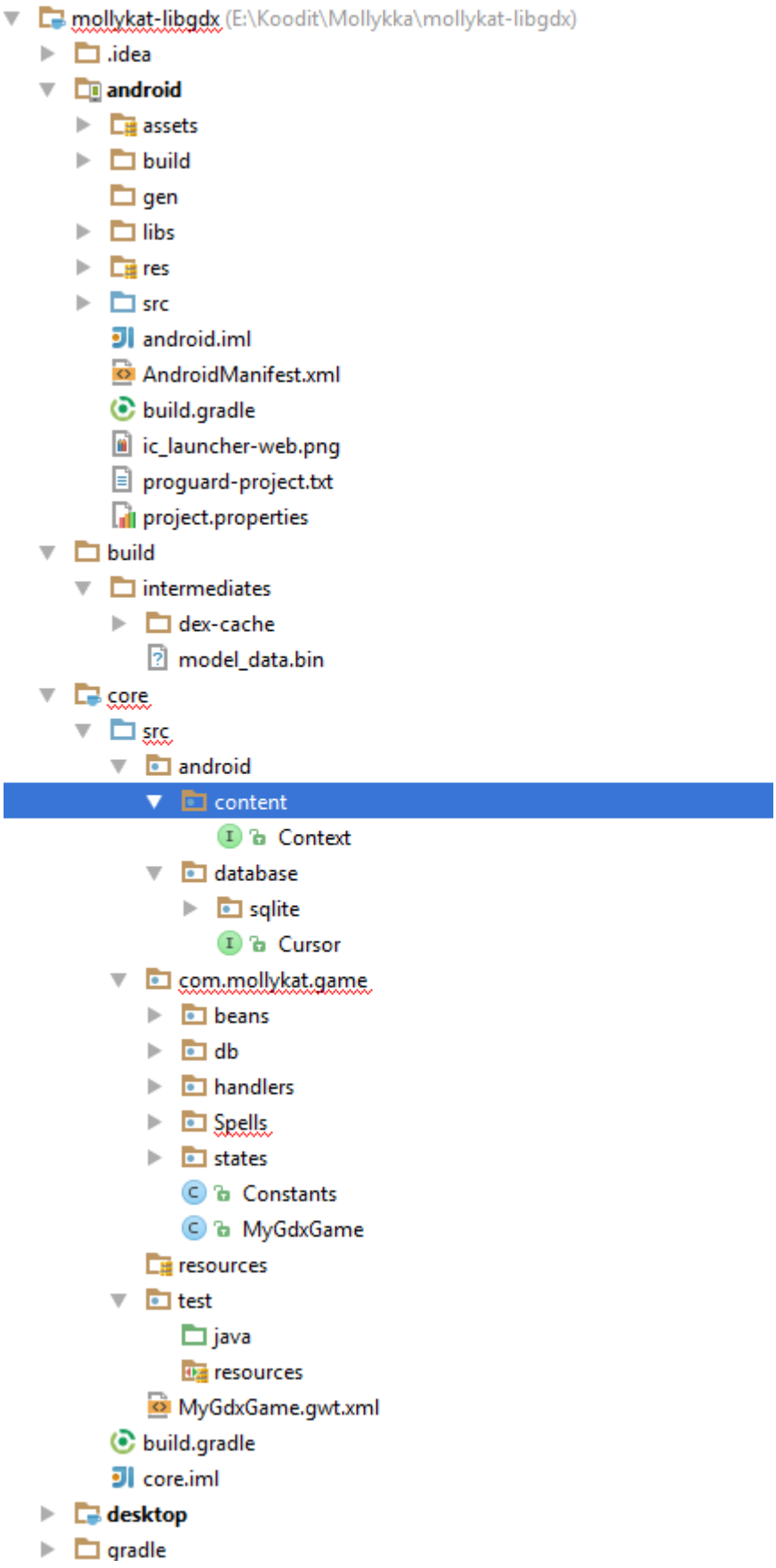
Päivitysten jakaminen tapahtuu Androidissa helposti Play Storen kautta. Kehittäjä uploadaa uuden version storeen ja Google jakaa sen eteenpäin laitteille jotka käyttävät kyseistä sovellusta. On tärkeää että sovellus ei hukkaa käyttäjän tekemiä asetuksia päivityksen yhteydessä.

5 ANDROID-OHJELMOINTI

Android-ohjelmoinnissa voidaan käyttää ohjelmointi ympäristönä voi käyttää mitä ohjelmaa itse haluaakaan, Google on tehnyt avoimen lähdekoodin Eclipsestä Androidille ADT bundle-nimisen paketin jota käytetään yleisesti. Toinen ohjelmointiympäristö mitä Google myöskin työstää on nimeltään Android Studio tosin tämä on vasta kokeiluversiossa.

5.1.1 Android kansiorakenne

Kun Android projekti luodaan siihen generoidaan automaattisesti ennalta määrätyt kansiot joista lopulta voidaan rakentaa .apk tiedosto, joka on valmis Android alustalla suoritettava ohjelma. Android projektin aloittaessa siihen generoidaan seuraavat tiedostot sekä kansiot:



Kuva Android sovelluksen kansio rakenteesta Android Studio ohjelmasta.

Src on kansio jonne sijoitetaan kaikki ohjelman lähdekoodit. Bin on käännettyjen sovellusten kansio eli siellä on .apk-tiedostot. Gen sisältää ADT:n automaattisesti luodut tiedostot**.** Assets kansio lähtökohtaisesti tyhjä kansio jonne voidaan sijoittaa raakatiedot. Res kansio sisältää usean alikansion, pääasiassa res kansiossa on tietoa jolla määritellään sovelluksen käyttöliittymää. Android käyttöliittymä määritellään xml tiedostoihin. Libs-kansio sisältää kolmannen osapuolen tuottamat kirjastot.

5.1.2 Ulkoasu

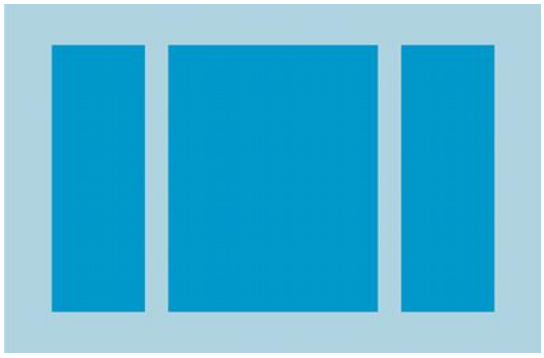
Ulkoasu toteutetaan XML tiedostona. XML tarkoittaa Extensible Markup Language se on metakieli jolla kuvataan tiedon rakennetta ilman ennalta määrättyjä koodeja (Wikipedia, XML 7.5.2015)

XML tiedoston rakenteen voi määritellä mm. seuraavalla tavalla:

```
<osasto>
	<tyontekija>
		<nimi>Matti Mallikas</nimi>
		<tyo>Graafikko</tyo>
		<palkka>1500</palkka>
	</tyontekija>
	<tyontekija>
		<nimi>Anna Testaaja</nimi>
		<tyo>Ohjelmoija</tyo>
		<palkka>1500</palkka>
	</tyontekija>
</osasto>
```

Tässä esimerkissä ”tyontekija” on osaston lapsi-elementti. Yhdessä osastossa voi olla useita työntekijöitä.

Android-ohjelmissa ulkoasu on aina toteutettu joko lineaarisesti, suhteellisesti, ruudukoilla tai listatyylisesti. Lineaarisessa ulkoasussa kaikki lapsi-elementit ovat joko vertikaalisesti tai horisontaalisesti sijoitettu yhteen putkeen kuten alla olevassa kuvassa.

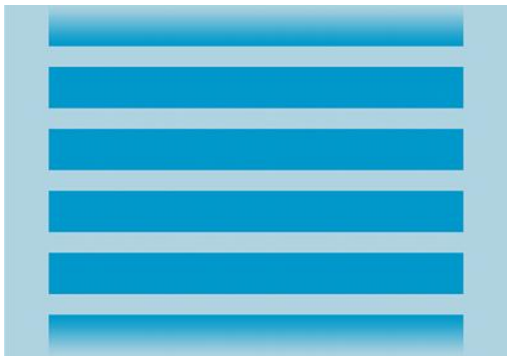


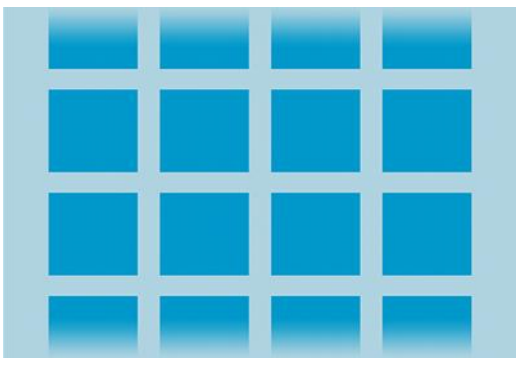
(viitattu http://developer.android.com/ 23.3.2014)

Suhteellisessa ulkoasussa voidaan kohdistaa lapsi-elementtejä kiinteisiin pisteisiin. Alla oleva kuva havainnollistaa suhteellisen ulkoasun toimintaa.

(viitattu http://developer.android.com/ 23.3.2014)

Viimeiset kaksi tapaa toteuttaa ulkoasu on lista näkymä joka on alaspäin vieritettävä vaihtoehto sekä ruudukko näkymä joka on kaksiulotteinen ruudukko, näistäkin toteutuksista esimerkki kuvat alla.





(viitattu http://developer.android.com/ 23.3.2014)

Mikään toteutus tapa ei ole tässäkään parempi kuin toinen vaan tulisi valita aina omaan käyttötarkoitukseen parhaiten sopiva vaihtoehto. Ulkoasun voi luoda itse alusta alkaen muokaten xml tiedostoa tai vaihtoehtoisesti käyttäen esim. Googlen Eclipse pluginista löytyvää työkalua ulkoasun luontiin. Tietenkin kaikki koodi mikä generoidaan automaattisesti, olisi suotavaa tarkistaa jälkeenpäin, jotta saadaan varmuus sen toimivuudesta.

6 PROJEKTI

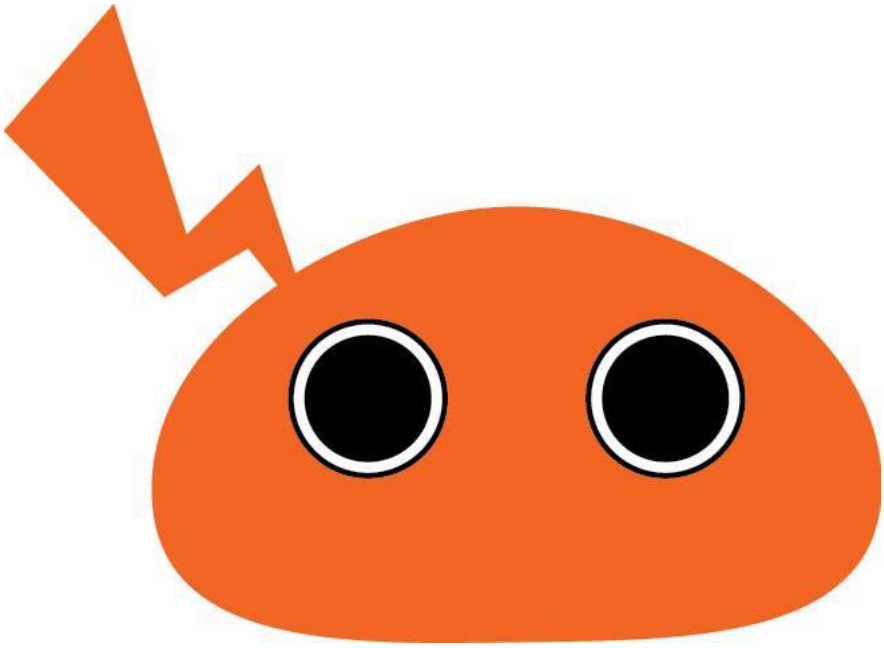
Sovelluksen kehittäminen lähti liikkeelle tarpeesta luoda idea. Halusimme osallistua mobiilipelillä kilpailuun. Alussa käytiin useita pitkiä keskusteluita muiden kehittäjien kanssa siitä mitä halusimme tehdä. Päätimme tehdä mobiilipelin projektinimeltään Möllykkä. Möllykkä nimi keksittiin vasta useita kuukausia myöhemmin. Peli idean kehittäminen alkoi tarpeesta tehdä peli ja samalla opinnäytetyö. Halusimme tehdä pelin mikä olisi eräänlainen koodaus osaamisen näyttö ja tekniikka demo. Päätimme silloin että pelissä ei tule olemaan mitään maata mullistavaa grafiikkaa sillä vielä tuolloin emme osanneet edes haaveilla että saisimme graafikoita tiimiin mukaan. Käytettävissä olevat resurssit ihmisten ja osaamisen osalta ovat olleet hyvin vaihtelevia koko projektin ajan. Siitä johtuen mahdollisen laadun arviointi on ollut haastavaa.

Lähdimme kehittämään peli-ideaa, peli jossa pelaaja voisi naksutella menupainikkeita ja samalla se koukuttaisi. Pitkän pohtimisen jälkeen päädyimme vuoropohjaiseen taistelu peliin. Pelin alussa jokainen pelaaja saa oman möllykän. Tämä möllykkä edustaa joko: Stamina, Strenght tai Intelligence tyyppiä. Möllykällensä voi kerätä kokemuspisteitä (Experience) ja näin ollen möllykän tasoa voi nostaa.

Maksimi tasoksi määritimme 25 jonka mukaan peli tullaan myös tasapainottamaan. Möllykät keräävät kokemuspisteitä taistelemalla toisten möllyköiden kanssa tai tekemällä pieniä tehtäviä.

Projekti päätettiin toteuttaa käyttäen Android Studiota sekä versionhallinnassa käytettiin Gittiä ja projektinhallinnassa Facebookin kehittämää Fabricaattoria. Projektia varten otettiin mukaan myös kaksi graafikkoa jotta peli saisi asianmukaiset grafiikat.

Alla graafikon konsepti kuva Möllykästä.



6.1 Taidot

Möllyköiden taidot päätettiin rajata niin että jokaisella Möllykällä löyty seuraavat taidot:

- perushyökkäys (Attack)
- kaksi perus tason taitoa mitkä tekevät vahinkoa
- yksi puolustautumiseen tarkoitettu taito (Defense)
- yksi erikois taito (Ultimate)

Päätimme jakaa skillit seuraaviin tyyppeihin:
Buff on skill tyyppi joka boostii hahmon toimintoja jollain tavalla. Esimerkkinä skill ”Apply Poison” jonka aktivoitua möllykän vuoro kuluu mutta sen jälkeisillä 4 kier-

roksella kyseinen möllykkä jättää DOTIN kohteeseen joka tekee X määrän vahinkoa per kierros.

Conjure eli lumous on skill tyyppi joka nostaa möllykän hyökkäys damagea suoraan. Esimerkki skillinä ConjureFire joka lisää tulityyppistä damagea hyökkäyksiin.
Protection eli suoja vähentää möllykän ottamaa vahinkoa X määrällä. Esimerkki skillinä FlameGuardian eli Tulisuojelija joka antaa sen loihtineelle möllykällä elämä pisteitä X määrän joka kierros sekä vähentää möllykän ottamaa tuli damagea kunnes se kuluu pois.

Vampiric skillit varastavat osan tehdystä damagesta itselle takaisin elämäksi. Vampiric tyypin skillejä on lähinnä Stamina möllyköillä. Vampiric skillistä annan esimerkkinä VampiricAuran joka on ultimate tason skilli mikä antaa kaikille skilleille vampiric kyvyn.

HOTS & DOTS eli heal over time sekä damage over time spellit ovat hyvin itseselittäviä. HOT skillit antavat elämää joka kierroksella ja DOT skillit taas vievät elämää joka kierroksella.

Tällä hetkellä skillejä on pelissä 34 kappaletta jotka jakautuvat kolmelle eri möllykkä tyypille. Skillien tasapainottaminen on pelinkehityksessä hyvin tärkeä vaihe. Siksi rakensimme sovelluksen joka käy jokaisen mahdollisen skill tyyppi combon lävitse ja pelaa sillä jokaista mahdollista skill tyyppi comboa vastaan satoja kertoja. Tavoitteenamme on saada n. 50 % voittomahdollisuus jokaiselle skill yhdistelmälle.

Oman möllykän voi myös lähettää tekemään jotain tehtävää missä kuluu aina X aikaa. Tänä aikana kyseistä möllykkää ei voi käyttää taistelussa. Ideana olisi koukuttaa pelaaja avaamaan möllykkä peli tietyin väliajoin saavutusten toivossa. Möllykässä tulee olemaan myös ingame shop missä mikromaksuja vastaan voi nopeuttaa tehtävien tekoa mutta ei kuitenkaan ostaa mitään mikä vaikuttaisi möllyköiden keskinäiseen taisteluun.

6.2 Toteutus

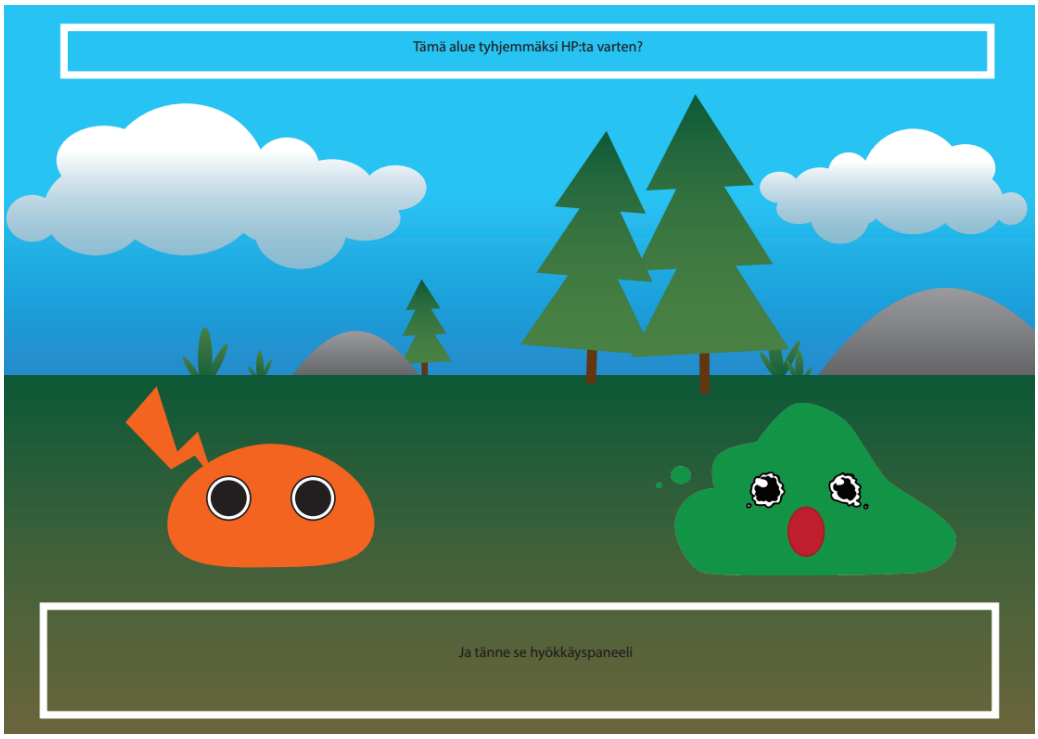
Viimeistelimme peli-ideaa puolivuotta ennen kuin aloimme koodaamaan riviäkään. Teimme muutamia UML-kaavioita helpottamaan toteutusta sitten kun sen aika tulisi. Pienenä miinuksena projekti eli myös toteutus vaiheessa joten aiemmin suunniteltuja UML-kaavioita ei ole oikeastaan hyödynnetty lainkaan.

Pelin koodaaminen alkoi keväällä 2014. Tuolloin lähdimme toteuttamaan Java kielellä PC:llä pyörivää sovellusta minkä tarkoituksena olisi helpottaa pelin tasapainottamista.

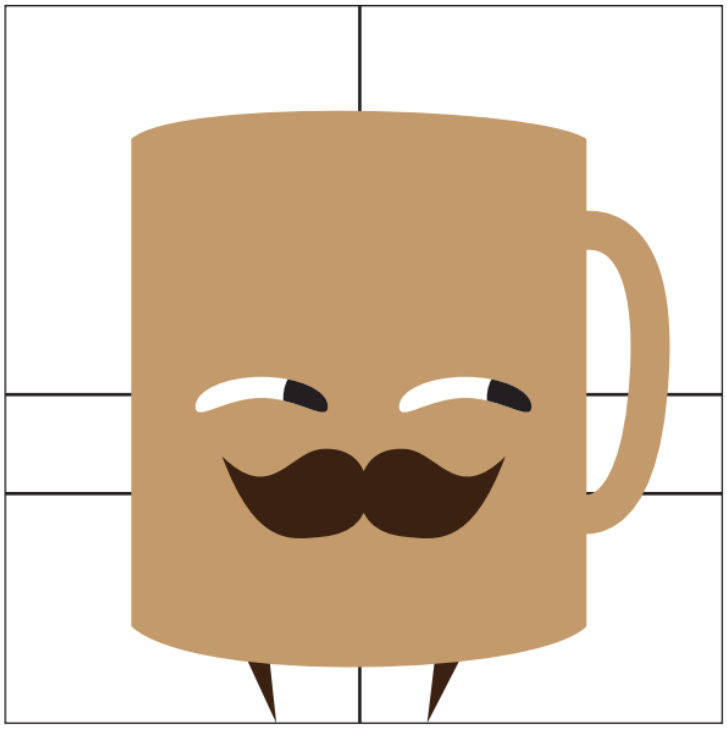
Sovellusta koodailtiin pitkin kesää. Ongelmaksi alkoi muodostua suoritustehojen puute koneissa tai liian tehottomat algoritmit tasapainotuksessa. Yksillä arvoilla tasapainotus sovelluksen suorittamiseen kului useita tunteja. Ajan puutteen takia tyydyimme jossain kohtaa saamiimme arvoihin millä ainoastaan muutama hahmo on ylivertainen toiseen nähden. Tarkoituksenamme on tasapainottaa peliä lisää myöhemmin esim. käyttäjä palautteen avulla.

Tulimme myös siihen tulokseen että peli tulee vaatimaan graafikoita. Saimme SAMK:in kautta 2 viestinnän opiskelijaa. He saivat hyvin nopeasti visioistamme kiinni ja lähtivät toteuttamaan peli grafiikkaa. Tällä hetkellä meillä on 3 ns. valmista möllykkää sekä 1 kenttä konsepti valmiina. Hahmojen muokattavuus lisää omaa haastettaan graafikoiden työssä.

Pelistä olemme rakentaneet version missä komentokehotetta käyttäen käyttäjä pystyy valitsemaan taidon mitä möllykkä käyttää ja sen vaikutukset lasketaan toisessa möllykässä. Käyttöliittymä on suunniteltu loppuun asti ja sen toteutus odottaa ainoastaan grafiikan valmistumista. Logojen ja fonttien valinnassa on esiintynyt hieman erimielisyyksiä.



Taistelu näkymä on alustavasti ylemmän kuvan mukainen. Käyttäjä näkee ala paneelissa omat skillinsä ja niiden recharge ajat. Yläpaneelissa vastaavasti näkyy aina vasemmalla oman möllykän HP bar sekä combo pisteet, oikealla puolella näkyy samat vastapuolelta. Skillin käyttämisen jälkeen sen tekemä vahinko tulee näkyviin keskellä ruutua jollain näyttävällä fontilla.



Jotta möllykät olisivat mahdollisimman paljon muokattavissa, niin teimme niiden silmistä sekä accessory palasista muokattavia. Jokainen möllykkä koostuu kehosta, silmistä sekä ei pakollisesta accessory osiosta. Aiemmassa kuvassa ylimääräinen osa on viikset. Käyttäjä pystyy valitsemaan omalle möllykälleen juuri sellaiset silmät kuin hän haluaa sekä juuri sellainen accessory palasen mikä hänestä tuntuu sopivimmallta möllykälleen.

Ajattelimme että hahmon muokkaus ominaisuus avautuu vasta kun on saanut möllykän kehitettyä tietylle tasolle, tästä on keskustelu edelleen käynnissä.

7 YHTEENVETO

Android ohjelman suunnittelu ja toteutus on valtava prosessi ja tämä työ oli vain pintaraapaisu asiaan. Java on kielenä helppo omaksua, mikäli hallitsee jonkin muun oliopohjaisen kielen esim. C# ja Android ohjelmoinnissa tärkeää on Android apin käyttäminen. Android apista on olemassa todella laadukas dokumentaatio missä on jopa esimerkkejä erilaisista toteutuksista.

Android on alustana myös houkutteleva sen suosion takia. Vaikkakin Javan kuolemaa on ennustettu pitkään, niin en epäile hetkeäkään ettei Android sovelluksen suunnittelu & toteutus olisi tällä hetkellä kannattavaa.

Itse projekti on edennyt hitaammin kuin olin odottanut. Nyt sovellusta on kehitetty 1,5 vuotta ja vastoinkäymisiä tulee jatkuvasti vastaan. Näin jälkeenpäin voin viisaampana todeta että projektit vaativat projektipäällikön jolla on valtaa ja auktoriteettia tehdä päätös jonka mukaan mennään. Meidän mallimme mukaan kaikki ovat samalla viivalla ja asioista keskustellaan niin kauan kunnes kaikki ovat samaa mieltä. Ideologisesti hieno juttu mutta näin käytäntöä seuranneena en suosittele tätä keinoa projektin ohjaamiseen kenellekään.

Hyviä puolia on ollut silti paljon, olen oppinut tämän projektin aikana merkittävästi uutta tietoa projektinhallinnasta, ohjelmoinnista ja Androidista. Projekti tulee jatkumaan näillä näkymin vielä useita kuukausia ennen kuin valmis peli saadaan Play Storeen. Tulevaisuuden suunnitelmissa on mm. peliäänten tuottaminen sekä lisä grafiikan saaminen peliin.

LÄHTEET

Git 2014 Viitattu 20.3.2014, http://git-scm.com/

Google 2014, Layouts Viitattu 23.3.2014
http://developer.android.com/guide/index.html

Google 2014, Android Studio Viitattu 23.3.2014
http://developer.android.com/sdk/installing/studio.html

Google 2014, ADT Bundle Viitattu 23.3.2014
http://developer.android.com/sdk/index.html

Grönholm, J. 2012, Ohjelmistotuotannon menetelmät. Luennot Satakunnan ammatti-korkeakoulun ohjelmistotuotannon kurssilla.

Katara M, Vuori M, Jääskeläinen A. 2013, Ohjelmistojen testaus. Viitattu 20.3.2014
http://www.cs.tut.fi/~testaus/s2013/luennot/TIE-21200_2013.pdf

Jönkkäri, T. 2007 Sekvenssikuvausjärjestelmän tietokannan ja julkaisujärjestelmän suunnittelu ja toteutus. Viitattu 20.3.2014
http://www.karhukamera.com/project/docs/opinnaytetyo.pdf

Wikipedia 2013, Vesiputous malli Viitattu 20.3.2014
http://fi.wikipedia.org/wiki/Vesiputousmalli

Wikipedia 2014, Ohjelmistotuotanto Viitattu 20.3.2014
http://fi.wikipedia.org/wiki/Ohjelmistotuotanto

Android ohjelman suunnittelu & Toteutus Jesse Kotiranta 2014 Viitattu 11.3.2015

Gameinformer 2012
http://www.gameinformer.com/b/features/archive/2012/12/27/tracking-the-popularity-of-videogame-genre.aspx Viitattu 29.4.2015