Haaga-Helia
ammattikorkeakoulu Oy

Katz - Android sovelluksen toteutus ja julkaisu

Nico-Petteri Miettinen

Opinnäytetyö
Tietojenkäsittelyn
koulutusohjelma
2015

Haaga-Helia

Tiivistelmä

Päiväys

Tekijä(t)
Nico Miettinen

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Katz – Android sovelluksen toteutus ja julkaisu

Sivu- ja liitesivumäärä
24

Opinnäytetyön otsikko englanniksi
Katz – development and publishing of an Android software

Opinnäytetyön tarkoituksena on toteuttaa peli Android laitteelle ja julkaista se Google Play kaupassa. Projektin ohjelmointikielenä käytetään Javaa ja työympäristönä käytetään Android Studiota.

Opinnäytetyössä kerrotaan projektin suunnittelusta, projektin rakenteesta, projektin toteutuksesta ja projektin julkaisusta. Suunnittelussa mietitään, mitä pitäisi suunnitella, ennen pelin toteutusta. Rakenteesta selviää projektin osa-alueet. Toteutuksessa katsotaan toteutusmenetelmiä eri osa-alueille. Julkaisussa selviää, miten sovelluksia voi julkaista Google Play kauppaan.

Opinnäytetyön lopussa käydään läpi tulokset. Tuloksien jälkeen pohditaan tuloksia ja aikaansaannosta. Pohditaan, mikä meni hyvin ja mitä voisi parantaa. Yhteenvedossa käydään opinnäytetyö lyhyesti vielä läpi ja pohditaan, mitä oppimiskokemuksia projektin aikana on tullut esille.

Opinnäytetyön kirjoittaja on ohjelmistokehittämiseen suuntautuva opiskelija. Pelien tekeminen on ollut kiinnostuksen kohde jo monta vuotta ja pientä kokemusta on kertynyt. Nämä kokemukset tukivat opinnäytetyön tekemistä. Projektin alussa uusia asioita oli Androidille ohjelmistokehitys, Android Studion käyttö ja julkaisu Google Play kauppaan.

Asiasanat
Android, ohjelmointi, mobiilipeli, julkaiseminen

Haaga-Helia
University of Applied Sciences

Abstract

Date

Author(s)
Nico Miettinen

Degree programme
Business Information Technology

Report/thesis title
Katz – development and publishing of an Android software

Number of pages and appendix pages
24

The purpose of this thesis is to implement a game for Android device and publish it on the Google Play store. This project's programming language is Java and the working environment is Android Studio.

The thesis goes through the project design, project structure, project development and project release. At the design we go through of what should be planned before developing the game. The structure tells about the different areas of the project. The development tells about the implementation of different areas of the project. The publication tells how applications can be published to Google Play store.

At the end of the thesis are the results. After that we reflect on the results. What went well and what could be improved. The summary goes through the thesis briefly and we reflect on what learning experiences the author has come across.

Thesis author is a software development student. Making games has been the object of interest for many years and the author has accumulated a little experience about developing games. These experiences supported the development of the project. At the beginning of the project, new things were development for Android, use of Android Studio software and publishing application at Google Play.

Keywords
Android, programming, mobile game, publishing

Sisällys

1 Johdanto....1
2 Teoriaosio....2
2.1 Java....2
2.2 Android....2
2.3 Android Studio....3
2.4 Open Game Art....3
2.5 Google Play....3
3 Projektin suunnittelu....4
4 Projektin rakenne....5
4.1 Manifest....5
4.2 Java....5
4.3 Assets....8
5 Toteutus....9
5.1 Pelin kuvaus....9
5.2 Toteutuksen yksityiskohdat....11
6 Testaus....19
6.1 Debuggaus....19
6.2 Käyttäjätestaus....19
7 Julkaisu Google Playssa....20
7.1 Projektin rakentaminen apk muotoon....20
7.2 Google Play kauppaan vienti....20
8 Tulokset....22
9 Pohdintaa....23
10 Yhteenveto....24
Lähteet....25

1 Johdanto

Tämän opinnäytetyön tarkoituksena on kertoa, millaisilla työkaluilla voi tehdä Android-käyttöjärjestelmälle pelin, minkälainen rakenne sillä voisi olla ja miten se käytännössä toteutetaan. Projektina toimii Katz peli, joka toteutettiin tätä opinnäytetyötä varten. Kyseinen projekti viedään myös Google Play kauppaan. Projekti toteutetaan Android-käyttöjärjestelmälle Javalla.

Mobiililaitteiden käyttö ja niillä pelaaminen on kasvanut hurjasti viimeisten vuosien aikana. Nykyään melkein jokaisella on jonkinlainen mobiililaite, oli se sitten älypuhelin tai tabletti (Cartesian). Android on Googlen julkaisema mobiilikäyttöjärjestelmä ja se on suosituin mobiilikäyttöjärjestelmä maailmassa (IDC). Sen avonaisuus helpottaa sovelluksien tekemistä Android laitteille.

Opinnäytetyön tekijä on kiinnostunut pelien kehittämisestä mobiililaitteille ja päätti tutkia prosessia. Lähinnä miten tämmöinen sovellus toteutetaan ja kuinka se julkaistaan Googlen Play kauppaan.

Mobiilikäyttöjärjestelmiä on muutamia. Suurimmat niistä ovat Googlen Android, Applen i-tuotteet ja Microsoftin Windows puhelin (IDC). Samoin koodauskieliä on monenlaisia. Tämän opinnäytetyön projekti on toteutettu Android laitteelle Javalla, koska opinnäytetyöntekijä on opiskellut pääosin Javaa Haaga-Heliassa ja hänellä on Android käyttöjärjestelmällinen puhelin ja tabletti. Tämä teki testaamisesta helpompaa.

Julkaisualustaksi valittiin Google Play, koska se on suurin Android kauppa. Se on myös Googlen virallinen sovellusten jako alusta.

2 Teoriaosio

Tässä luvussa käsitellään teoriataustaa tekniikoita ja työkaluja, joita projektissa käytettiin. Projektin tekemiseen käytettiin pöytäkonetta, jonka käyttöjärjestelmänä on Windows 8.1. Pelin alustana toimii Android käyttöjärjestelmä ja peli ohjelmoitiin Javalla. Työympäristönä käytettiin Android Studiota. Resurssit kerättiin Open Game Artista.

2.1 Java

Javaa käytetään projektin ohjelmointikielenä. Java on Sun Microsystemsin kehittämä oliopohjainen ohjelmointikieli ja se on maailmanlaajuisesti standardi mobiilisovelluksille, peleille, nettipohjaisille sisällöille ja yrityssovelluksille. Java on suunniteltu alustapohjattomaksi, mahdollistaen ohjelmien tekemisen käyttöjärjestelmille ilman niiden kääntämistä kyseiselle järjestelmälle. (Oracle)

Javan koodi kirjoitetaan ensin tekstiversiona java tiedostoon. Sen jälkeen se kootaan class tiedostoiksi. Nämä tiedostot eivät sisällä koodia, joka on natiivi prosessorillesi. Sen sijaan ne sisältävät bittikoodeja, jotka ovat Java Virtual Machinen (Java VM) koodikieltä. Koska Java VM on saatavilla monilla eri käyttöjärjestelmillä, samat class tiedostot pystytään lukemaan eri käyttöjärjestelmillä. (Gallardo, R, Hommel, S, Kannan, S, Gordon, J & Zakhour, S 2015)

2.2 Android

Projekti toteutettiin Androidille. Android on avoimeen lähdekoodiin perustuva ohjelmistokasa johon kuuluu käyttöjärjestelmä. Sen perustana on UNIX käyttöjärjestelmä. Android suunniteltiin alusta alkaen mobiililaitteille. (Google b)

Androidista on tulossa kokoajan suositumpi mobiilimarkkinoilla. Android pohjautuu avoimeen lähdekoodiin, jonka lisäksi osa kehittämistyökaluista on ilmaisia, joten sovelluksia kehitetään jatkuvasti. Tämä kannustaa suuresti ihmisiä valitsemaan Androidin. Androidin sovellukset ovat Java pohjaisia ja tämä merkitsee virtuaalikoneen ympäristönkäyttöä, hyötyineen ja haittoineen. (Sharma, M & Thakur, A 2015)

Androidin kehitti Android Inc, joka perustettiin lokakuussa 2003 Palo Altossa Kaliforniassa. Firman perustajiin kuuluivat Andy Rubin, Rich Miner, Nick Sears ja Chris White. Tarkoituksena oli kehittää älykkäämpi mobiililaite, joka olisi enemmän tietoinen käyttäjän sijainnista ja mieltymyksistä. Google osti Android Oy:n elokuussa 2005. (Sharma, M)

2.3 Android Studio

Projekti toteutettiin Android Studiolla. Android Studio on kehitysympäristö (IDE), joka on tarkoitettu Android sovelluksien tekemiseen. Android Studio perustuu IntelliJ IDEA kehitysympäristöön. (Google a)

Alun perin Eclipse Android Develpment Tools oli Googlen pääkehitysympäristö Android sovelluksien kehittämiseen mutta se vaihdettiin Android Studioon. Ensimmäinen stabiili versio julkaistiin joulukuussa 2014.(Protalinski, E. 2014)

2.4 Open Game Art

Open Game Art on sivusto, joka jakaa ilmaislisensseihin perustuvaa materiaalia pelinkehittäjille. Sivusto sisältää mm. 2D kuvia, 3D malleja, musiikkia, ääniä, tekstuureita yms. Sivuston tarkoituksena on antaa kehittäjille ilmaista materiaalia, jotta kehittäjien projektit eivät käyttäisi ”placeholder” materiaaleja lopullisessa versiossa. Sivusto perustuu pelien kehittäjien yhteisön ylläpitämiseen ja materiaalin jakamiseen. (Open Game Art)

2.5 Google Play

Google Play on Googlen virallinen kauppa Android käyttöjärjestelmälle. Google Play sisältää miljoonia sovelluksia, pelejä, e-kirjoja, musiikkeja, elokuvia yms. Google Playn kautta ostamasi tuotteet tallentuvat pilveen ja voit käyttää niitä kaikilla Android laiteillasi. Alun perin Googlen kaupan nimi oli Android Market. (Cutlack, G 2012)

3 Projektin suunnittelu

Project Katz on mobiilipeli, joka toimii Android käyttöjärjestelmällä ja peli toimii Android SDK versiolla 15 ja siitä ylöspäin. Tähän syynä on se, että projektin tarkoitus on toimia mahdollisimman monella alustalla. Peli on koodattu Javalla, koska Androidissa on Javalle natiivituki. Pelityypiksi valittiin roolipeli, koska kehittäjää itseään kiehtoo roolipelit niiden yleisen fantasiateeman ja pelattavuuden vuoksi.

Projekti alkaa ideasta, jota lähetään viemään eteenpäin. Idea voi olla aluksi pieni ja vähän hölmön tuntuinen mutta kun sitä viedään eteenpäin, alkaa se näyttämään hyvältä ja järkevältä.

Tämän projektin idea alkoi halusta tehdä hyvin pieni roolipeli. Yksinkertainen mutta samalla vuorovaikutteinen. Pelaajan tarkoitus olisi taisteluissa interaktiivisesti väistellä vihollisen iskuja ja samalla iskeä viholliseen. Kun pelaajan tai vihollisen elämäpisteet tippuisivat nollaan, tappelu päättyisi. Tappeluun piti suunnitella erilaisia hyökkäyksiä, jotta se ei olisi todella yksitoikkoinen. Hyökkäyksille suunniteltiin erilaisia liikeratoja, jotta pelaajan olisi vaikeampi reagoida niihin. Tämä nostaisi pelin vaikeustasoa mitä pidemmälle pelissä päästäisiin.

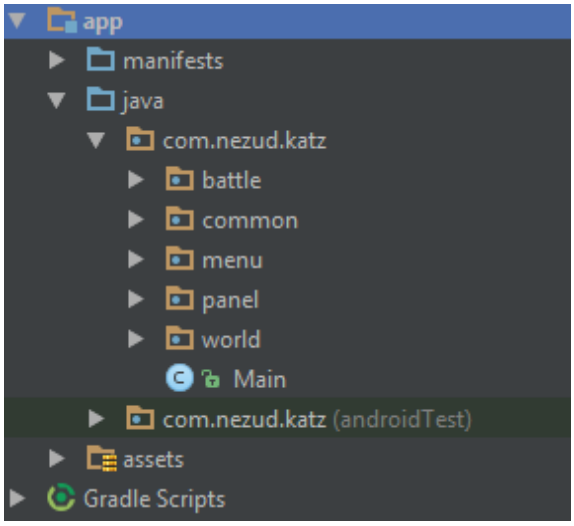
Pelkkään tappeluun kohdistuva peli voisi tuntua hieman tylsältä. Niinpä piti suunnitella jokin muu alue peliin mukaan. Useissa roolipeleissä on taisteluiden ulkopuolella ulkomaailma. Tässä pelaaja liikkuisi eteenpäin ja tarina etenisi. Tästä ideasta lähdettiin liikkeelle. Mutta pelin yksinkertaisuuden vuoksi, ulkomaailmastakin pitäisi tehdä yksinkertainen. Niinpä suunnitelmissa peliin tehtiin ulkomaailma, joka etenisi itsestään mutta samalla siinä olisi dialogeja, jotta tarina voisi edetä.

Pelin päättäminen suunniteltiin viimeiseen pomotaisteluun. Tämä eeppinen yhteenotto pelaajan ja viimeisen pomon kanssa toisi tarinan päätökseen.

Useissa peleissä on jonkinlainen tarina. Tarina antaa pelaajalle motivaatiota pelata peli läpi, jotta tarina saisi päätöksen. Peliin suunniteltiin hyvin yksinkertainen tarina, jossa pahis vie päähahmolta jotain hänelle tärkeätä. Tämä antaa päähahmolle motivaation lähteä etsimään pahista.

4 Projektin rakenne

Projektin osat on eritelty eri osioihin, jotta projektia olisi helpompi hallita ja jotta se olisi siisti. Projektin rakenteesta selviää projektin pääalueet ja missä mikäkin ominaisuus sijaitsee. Tässä luvussa kerrotaan projektin ominaisuuksista ja luokista. Nämä avaavat pelin kehittäjille luokkien mahdollisesta rakenteesta.



Kuva 1. Rakenne

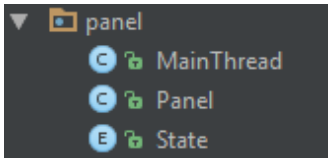
4.1 Manifest

Manifest pitää sisällään projektin manifest tiedoston, joka antaa projektille ominaisuudet kuten projektin nimi, kuvake, mahdollinen teema ja kuvan suunta. Tässä projektissa manifest tiedostoon vietiin uusi kuvake ja vaihdettiin kuvan suunta vaakatasoon.

4.2 Java

Java sisältää projektin kaiken javalla tehdyn koodin. Projekti on toteutettu javalla, joten tämä kansio sisältää itse pelin.

4.2.1 Paneeli (Panel)

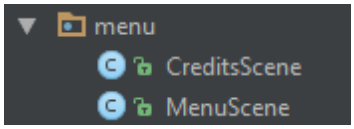


Kuva 2. Paneelin luokat

Paneeli on näyttö, joka näytetään kun sovellus avataan eli käytännössä paneeli näyttää käyttäjälle kaiken mitä pelissä tapahtuu. Tämä myös tarkoittaa sitä, että paneeli pyörittää peliä ja kaikkea, mitä siihen kuuluu.

MainThread on pelin päälanka, joka pyörittää peliä. Tässä luokassa lasketaan pelin frames per second, ja yritetään pitää se kuudessakymmenessä, eli peli yrittää pyörittää koodia 60 kertaa sekunnissa. Panel on pelin paneeli, jossa kaikki koodi pyörii. State on pelin tila, joka määrittää, mitä pelin osaa pyöritetään. Esimerkiksi kun tila on BATTLE, niin peli pyörittää taistelua.

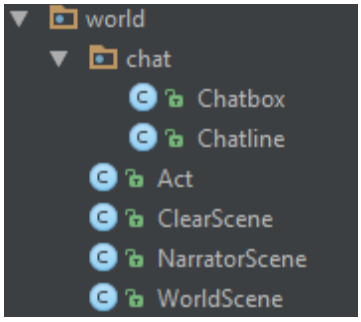
4.2.2 Menu



Kuva 3. Menun luokat

Menu sisältää pelin päävalikon ja tekijät. MenuScene pyörittää pelin päävalikkoa ja CreditsScene pyörittää pelin tekijät valikkoa.

4.2.3 World



Kuva 4. Worldin luokat

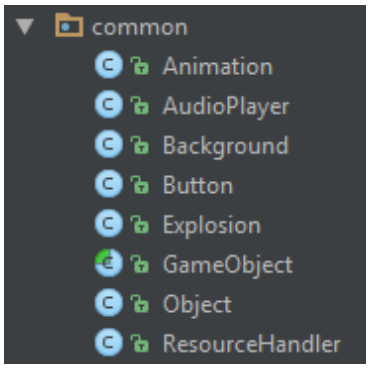
World (maailma) sisältää pelin ulkomaailman ja siihen kuuluvat luokat. Worldissa oleva WorldScene luokka pyörittää maailmaa. Worldia kutsutaan paneelista kun pelin tila on WORLD. Act on kenttien yksi taso, jossa on vihollinen. Yleensä kentissä on kolme tasoa. Chatbox on dialogin laatikko, jossa pyöritetään dialogiin liittyvää koodia. Chatline on aina yksi linja. ClearSceneä pyöritetään joka kentän jälkeen, eli tämä on tason läpäisy valikko. NarratorScene pyöritetään kun kertoja kertoo tarinasta. Pelissä kertoja esiintyy kaksi kertaa: pelin alussa ja pelin lopussa.

4.2.4 Battle

Kuva 5. Battlen luokat

Battle (taistelu) sisältää pelin taistelun ja siihen kuuluvat luokat. Battlessa oleva BattleScene luokka pyörittää taistelua. Battlea kutsutaan WorldScene luokasta kun pelin tila on BATTLE. BattleType määrittää tappelun tyypin, eli onko kyseessä normaalitappelu vai pomotappelu yms. Enemy on vihollinen tappelussa ja se sisältää viholliseen liittyvän koodin. Player sisältää pelaajan ja siihen liittyvän koodin. Fade on taistelun alussa tuleva himmennys, jossa ilmoitetaan, että taistelu on alkanut. STB eli ”Status Text Bar” sisältää käyttöliittymään liittyvän informaation eli pelaajan elämäpisteet, hyökkäysmittarin, parannusmittarin ja vihollisen elämäpisteet.

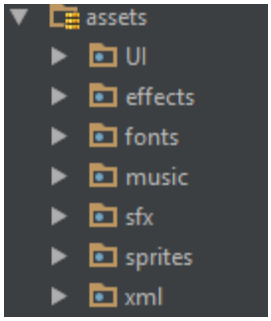
4.2.5 Common



Kuva 6. Commonin luokat

Common pitää sisällään kaikki yleiset luokat, joita käytetään monessa eri osassa peliä. Animation on pelin animaatio luokka. AudioPlayer pyörittää pelin äänieffektejä ja musiikkia. Background on kenttien ja taisteluiden tausta. Buttonia käytetään pelin nappuloihin. Explosion on vihollisen ja pelaajan räjähdys. GameObject on pelin abstrakti luokka, jota käytetään melkein kaikissa pelin objekteissa (pelaaja, vihollinen tausta yms.). Tämä luokka sisältää x ja y akselin koordinaatit, leveyden ja pituuden ja kuvan. Object on yleinen luokka, jolla ei ole mitään erikoisia ominaisuuksia ja sitä käytetään yksinkertaisiin ratkaisuihin kuten palkin tekemiseen, jotta pelaaja ei putoaisi kentästä läpi tai valikon piirtämiseen. ResourceHandler käsittelee kaikki pelin resurssit. Sillä ladataan mm. pelien kentät ja niihin liittyvät tiedot XML tiedostoista, kuvat, äänet ja musiikit.

4.3 Assets



Kuva 7. Materiaalit

Assets sisältää projektin materiaalit. Materiaaliin kuuluu xml tiedostot, joista ladataan kentät, viholliset, dialogit ja hyökkäykset, grafiikat ja animaatiot, fontit, musiikit ja äänet. Materiaalit on jaoteltu kansioihin sen mukaan, mihin kokonaisuuteen ne kuuluvat. Esimerkiksi musiikit on laitettu music nimiseen kansioon.

5 Toteutus

Peli on rakennettu kokonaan tyhjältä pohjalta, eli tässä ei ole käytetty muiden tekemiä rajapintoja. Hyötypuolena tässä on ymmärrys koko projektin toimintaan ja sen hallintaan mutta haittana taas on aikaa vievä osuus kun projekti pitää rakentaa tyhjästä.

5.1 Pelin kuvaus

Peli keskittyy lähinnä kahteen osa-alueeseen: taisteluun ja ulkomaailmaan. Taistelu on itse pelin suola, jossa varsinainen pelaaminen tapahtuu, ja ulkomaailma luo pohjaa taisteluille.

5.1.1 Tarina

Pelin tarina on hyvin yksinkertainen. Pimeyden herra on vienyt päähahmolta, jonka nimi on Katz, hänen maagisen appelsiinin. Tämän jälkeen koko saari, jolla Katz asuu, on mennyt kaaoksen partaalle. Hirviöitä on joka puolella ja kaikki ovat alkaneet olemaan väkivaltaisia. Katzin tehtävänä on hankkia appelsiini takaisin ja tuoda rauha saarelle.

5.1.2 Taistelu

Taistelut alkavat, kun kohtaat vihollisen ulkomaailmassa. Kuvaruudulla näkyy kenttä, ohjattava hahmo ja vihollinen. Samalla näkyy hahmon elämäpisteet, hyökkäysmittari ja parannusmittari sekä vihollisen elämäpisteet ja hyökkäysmittari. Tarkoituksena on kukistaa vihollinen samalla kun pelaaja väistelee vihollisen hyökkäyksiä.

Kuva 8. Taistelu

Attribuutit määrittelevät, kuinka voimakas pelaajan hahmo on. Hahmolla on viisi eri attribuuttia:

- Elämäpisteet: määrittelevät kuinka monta osumaa pelaaja voi ottaa, ennen kuin häviää pelin. Jos elämäpisteet tippuvat nollaan, on pelaaja hävinnyt.
- Voima: määrittelee kuinka voimakkaita pelaajan iskut ovat.
- Hyökkäysmittari: määrittelee milloin pelaaja voi hyökätä.
- Parannus: määrittelee kuinka voimakas pelaajan parannuskyky on.
- Parannusmittari: määrittelee milloin pelaaja voi käyttää parannusta.

Kenttä määräytyy ulkomaailman mukaan. Itse kentällä ei ole niinkään merkitystä taistelun kannalta mutta taistelun tausta muuttuu kentän mukaan.

Sekä pelaajalla, että vihollisella on oma hyökkäysmittari. Kun pelaajan hyökkäysmittari on täynnä, voi pelaaja painaa hyökkäysnappulaa hyökätäkseen. Hyökkäykset tekevät vahinkoa pelaajan voiman ja iskun kertoimen verran. Pelaajalla on kolme eri hyökkäystä:

- hidas hyökkäys, jonka kerroin on suuri
- normaali hyökkäys, jonka kerroin on yksi
- nopea hyökkäys, jonka kerroin on pieni

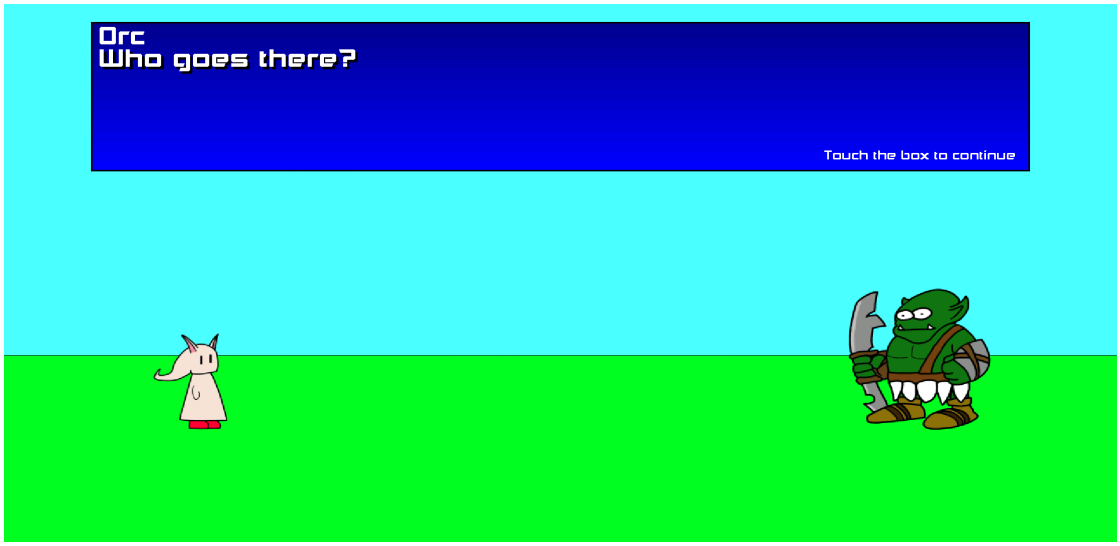
Vihollisella hyökkäys tapahtuu kun vihollisen hyökkäysmittari on täynnä. Vihollinen valitsee hyökkäyksen omasta iskuvalikostaan satunnaisesti. Hyökkäykset tekevät vahinkoa vihollisen voiman ja iskun kertoimen mukaan.

Pelaaja voi väistää vihollisen hyökkäyksiä joko hyppäämällä tai olemalla paikoillaan. Hypätä voi näpäyttämällä ruutua. Aluksi hyökkäykset tulevat vain suoraan kohti mutta pidemmälle peliä päästäessä vihollisten iskut muuttuvat monimutkaisemmiksi ja saattavat jopa osua pelaajaan useampaan kertaan.

Parantaminen palauttaa pelaajan elämäpisteitä. Kun parannusmittari on täynnä, voi pelaaja painaa parannusnappia parantaakseen hahmoaan.

5.1.3 Ulkomaailma

Ulkomaailmassa pelaaja kohtaa viholliset ja keskustelee heidän kanssaan ennen tappeluun siirtymistä. Pientä sanaharkkaa tapahtuu ja tarina etenee.



Kuva 9. Ulkomaailma

Kun pelaaja on voittanut kaikki kentän viholliset, kenttä määritellään läpäistyksi. Kentän läpäisy antaa pelaajan kehittää hahmoaan valitsemalla yhden kolmesta attribuutista:

- Elämäpisteet: kasvattaa elämäpisteitä viidellä kymmenellä.
- Voima: kasvattaa voimaa viidellä pisteellä mutta samalla kasvattaa hyökkäysmittaria 50:llä pisteellä.
- Parannus: kasvattaa parannusta 15:llä pisteellä mutta samalla kasvattaa parannusmittaria 100:lla pisteellä.

5.2 Toteutuksen yksityiskohdat

Projekti toteutettiin kotikoneella, jossa oli Windows 8.1. Ohjelmointiin käytettiin Android Studiota ja ohjelmointikielenä käytettiin Javaa.

5.2.1 Valikko

Pelin valikosta tehtiin hyvin yksinkertainen. Valikossa lukee pelin nimi ja valinta vaihtoehtoina on pelin aloittaminen, ohjeet ja poistuminen. Ensiksi aloitettiin luomalla valikolle oma luokka. Luokka koodattiin ensimmäisenä näyttämään pelin nimi. Sitten luokkaan koodattiin valinnat nappuloina. Kuvasta 10 näkyy luokan rakennusmetodi ja siihen kuuluvat muuttujat. Pelin aloitus koodattiin vaihtamaan pelin tila WORLD:ksi. Ohje koodattiin näyttämään usean sivun ohjekirja, josta voi painamalla mennä sivuja eteenpäin ja taaksepäin. Lopuksi koodattiin pelistä poistuminen.

```
public MenuScene(Panel panel){
    bg = new Background();
    start = new Button(350, 500, 150, 100, "Start");
    help = new Button(575, 500, 150, 100, "Help");
    quit = new Button(WIDTH - 350 - 150, 500, 150, 100, "Quit");

    backSize = 40;
    moved = 0;
    curPage = 0;
    helpPages = 5;

    back = new Object[HEIGHT / backSize + 1][WIDTH / backSize + 1];

    createBack();

    backPanel = new Object(WIDTH / 2 - 400, 100, 800, 550);
    next = new Button(backPanel.getX() + backPanel.getWidth() - 180, backPanel.getY() + backPanel.getHeight() - 130, 150, 100, "Next");
    prev = new Button(backPanel.getX() + 10, backPanel.getY() + backPanel.getHeight() - 130, 150, 100, "Previous");
    removeHelp = new Button(help.getX(), backPanel.getY() + backPanel.getHeight() - 130, 150, 100, "Return");

    this.panel = panel;

}
```

Kuva 10. MenuScenen rakennusmetodi

5.2.2 Ulkomaailma

Ulkomaailman koodaaminen aloitettiin luomalla luokka nimeltä WorldScene. Tämän tarkoitus on pitää sisällään kaikki ulkomaailman pyörittämiseen liittyvä koodi. Tämän jälkeen luotiin ulkomaailmaan pelaaja ja vihollinen. Tämän jälkeen pelaajalle koodattiin liikkuminen tiettyyn pisteeseen asti. Kun pelaaja saavuttaa tietyn pisteen x-akselilla, pelaaja pysäytetään. Tämän jälkeen koodattiin tekstilaatikko, jossa näkyy dialogi. Tekstilaatikko valuu ruudun yläpuolelta tiettyyn pisteeseen asti. Tämän jälkeen näytetään dialogi. Teksti tulee nopeasti kirjain kerrallaan ruudulle näkyviin. Kun pelaaja painaa tekstilaatikkoa, niin dialogi etenee. Viimeisen puheenvuoron jälkeen luodaan taistelu ja vaihdetaan pelin tilaksi BATTLE. Taistelun jälkeen vaihdetaan pelin tilaksi taas WORLD. Vihollista ja tekstilaatikkoa ei enää piirretä ja pelaajan hahmo liikkuu ruudulla eteenpäin, kunnes pelaaja on siirtynyt ruudun ulkopuolelle. Tämän jälkeen ladataan uusi vihollinen.

Kun kaikki viholliset on voitettu, vaihdetaan pelin tilaksi CLEAR ja pelaaja siirtyy tason voittamisruutuun. Tämän koodaamiseen ei mennyt hirveästi aikaa, koska ruutu on hyvin yksinkertainen. Pelaaja voi valita kolmesta attribuutista, yhden jota haluaa kasvattaa. Sen jälkeen kun pelaaja on valinnut attribuutin, tulee hänen painaa ruutua vielä kerran. Kuvasta 11 näkyy kosketusmetodi, jolla pelaajan kosketuksen mukaan tehdään lisätään pelaajan haluama attribuuttia. Tämän jälkeen ladataan uusi kenttä.

```
public void onTouch(int tX, int tY){
    if (attribute > 0){
        if (tank.getRectangle().contains(tX, tY)){
            hp += hpIncrease;
            hpMax += hpIncrease;
            attribute--;
            playSound("Stat Up");
        }else if(dps.getRectangle().contains(tX, tY)){
            power += powerIncrease;
            attack += attackIncrease;
            attribute--;
            playSound("Stat Up");
        }else if(healer.getRectangle().contains(tX, tY)){
            heal += healIncrease;
            healGauge += healGaugeIncrease;
            attribute--;
            playSound("Stat Up");
        }
    }else{
        playSound("Select");
        panel.nextScene();
        attribute = attributeIncrease;
        musicPlaying = false;
    }

}
```

Kuva 11. ClearScenen kosketus metodi

Aivan lopuksi kenttiin koodattiin kertojan oma osa. NarratorScene pyörittää kertojaa, jossa on musta tausta ja valkoisella tekstillä lukee kertojan dialogi. Kertojaa käytetään pelin alussa, jotta pelin tarinalle voidaan luoda pohja. Kertojaa käytetään myös pelin lopussa, jossa kerrotaan pelin lopputilanne. Tässä käytetään hyväksi Dialogia, joka koodattiin kenttiin, joten tästä luokasta tuli hyvin yksinkertainen. Tason luokan koodi pystyttiin melkein kokonaan uudelleenkäyttämään NarratorScene luokan luomisessa.

5.2.3 Taistelu

Taisteluun meni selvästi eniten aikaa. Taistelu on kaikista osa-alueista monimutkaisin mutta ei mitenkään mahdoton. Taistelu aloitettiin luomalla BattleScene luokka, joka sisältäisi kaiken taisteluun kuuluvan koodin.

Tämän jälkeen aloitettiin luomalla pelaaja. Pelaajan ominaisuuksiin eritoten kuuluu hyppy. Hyppy toimii liikuttamalla pelaajaa y-akselilla. Jotta pelaaja ei putoisi ruudusta ulos, piti luoda maataso, johon pelaajan osuttua hahmo lopettaisi putoamisen. Pelaajan ominaisuuksiin kuuluvat myös elämäpisteet, voima, hyökkäysmittari, parannus ja parannusmittari. Kuvasta 12 näkyy pelaajan rakennusmetodi, jolla annetaan pelaajalle

ominaisuuksia. Hyökkäysmittaria ja parannusmittaria päivitetään pelaajan mukana. Joka kerta, kun pelaajan päivitysmetodia kutsutaan niin hyökkäs- ja parannusmittari kasvavat yhdellä.

```
public Player(int hp, int maxHp, int power, int att, int heal, int healG){
    super.x = PX;
    super.y = PY;
    super.width = 60;
    super.height = 100;

    this.health = hp;
    this.healthMax = maxHp;
    this.power = power;
    this.attGaugeMax = att;
    this.heal = heal;
    this.healGaugeMax = healG;

    jumpPower = -13;
    gravity = 0.5f;

    flashDur = 50;
    flash = 0;
    flashing = false;
    immune = false;

    active = true;

    bottom = new Object(x, y + height + 1, width, 1);

    sheet = res.fetchImage("Katz", "sprites");

}
```

Kuva 12. Pelaajan rakennus metodi

Vihollinen luotiin pelaajan jälkeen. Vihollisen ei tarvitse osata hypätä, joten sille ei tarvinnut erikseen koodata hyppyä. Muuten vihollisella on melkein samat ominaisuudet kuin pelaajalla. Tämän vuoksi vihollisen pystyi melkein kokonaan luomaan pelaajaluokan pohjalta.

Iskut koodattiin seuraavaksi. Iskuille koodattiin kolme eri tyyppiä: suora, kaari ja aalto. Suora lentää suoraan kohti. Kaari lentää kaaressa ja sille voi määritellä, kuinka jyrkän kaaren isku tekee. Aalto lentää suoraan aaltoliikkeenä. Jotta iskuja voitaisiin kutsua yhdellä luokalla, piti iskuille koodata abstrakti luokka, josta kaikki iskut ottavat ominaisuuksia itselleen. Kuvasta 13 näkyy abstraktiluokan muuttujat. Iskuja kutsutaan taistelussa kun pelaaja tai vihollinen on valinnut itselleen hyökkäyksen. Pelaaja ei voi hypätä samanaikaisesti hyökätessään, mutta pelaaja voi hypätä kun vihollinen hyökkää. Jos iskut osuvat, menettää osuman saanut osapuoli vastustajan voiman ja iskun kertoimen verran elämäpisteitä. Kuvasta 14 näkyy pelaajan hyökkäysmetodi.

```
public abstract class Attack extends GameObject implements Cloneable {

    protected int repeat;
    protected int curPattern = -1;
    protected double multiplier;
    protected Order order;
    protected LinkedList<Pattern> patterns;
```

Kuva 13. Attack on abstraktiluokka kaikille hyökkäyksille

```
//Attack update
try {
    pAtt.update();
    if (pAtt.getX() > Panel.WIDTH || pAtt.getY() > Panel.HEIGHT) {
        pAtt = null;
        p.setAttacking(false);
    }
    if (collision(pAtt, e) && !e.isImmune()) {
        e.subHealth((int) (p.getPower() * pAtt.getMultiplier()));
    }
} catch (Exception e) {
}
```

Kuva 14. Pelaajan hyökkäykseen liittyvä koodi

Sitten koodattiin graafinen käyttöliittymä, jotta pelaaja tietäisi, paljonko hänellä on elämäpisteitä, milloin hän voi hyökätä ja parantaa ja kuinka paljon vihollisella on elämäpisteitä jäljellä. Tähän käytettiin luokkaa nimeltä STB eli Status Text Bar. Kuvasta 15 käy ilmi STB luokan rakennusmetodi ja siihen kuuluvat muuttujat.

```
public STB(int x, int y, String text, int current, int max, int c, boolean showValues){
    super.x = x;
    super.y = y;
    super.width = 250;
    super.height = 20;
    this.current = current;
    this.max = max;
    this.text = text;
    this.color = c;
    this.showValues = showValues;

    setFullText();

}
```

Kuva 15. STB:n rakennus metodi

Seuraavaksi koodattiin voitto ja häviö. Kun vihollisen elämäpisteet tippuvat nollaan, räjähtää vihollinen ja pelaajalle ilmoitetaan voitosta. Kuvasta 16 näkyy voittotilanteen käsittely. Samoin jos pelaajan elämäpisteet tippuvat nollaan, pelaajan hahmo räjähtää ja pelaajalle ilmoitetaan häviöstä. Voittotilanteessa koodattiin jatkaminen painamalla ruutua. Hävitessä kysytään, haluaako pelaaja yrittää uudestaan vai palata valikkoon. Vaihtoehdot näytetään pelaajalle nappuloina. Uudelleenyritys aloittaa tappelun alusta samoilla

elämäpisteillä, kuin tappeluun saapuessa ja valikkovaihtoehto palauttaa pelaajan takaisin päävalikkoon.

```
//Victory condition
if (victory){
    if (e.getType() == BattleType.FINAL) {
        e.removeFlash();
        stopMusic();
        victoryEvent();
    }
    e.getExplosion().update();
}
```

Kuva 16. Voittokoodi

5.2.4 Tekijät

Pelin loppuun koodattiin tekijät. CreditsScene luokkaa käytetään tekijöiden nimien näyttämiseen. Kuvasta 17 käy ilmi, kuinka tekijät piirretään näytölle. Tekijät koodattiin liikumaan yksi osa-alue kerrallaan ruudussa. Nimet liikkuvat ylöspäin ruudun alalaidasta käyttäen y-akselia. Kun kaikki tekijät on näytetty, koodattiin lopuksi näkyviin teksti ”The End” kuvaamaan pelin läpäisyä. Lopulta koodattiin mahdollisuus palata päävalikkoon painamalla ruutua.

```
public void render(Canvas c){
    Paint paint = new Paint();
    paint.setTextAlign(Paint.Align.CENTER);
    //bg.renderDebug(c);
    paint.setColor(Color.BLACK);
    c.drawRect(0, 0, WIDTH, HEIGHT, paint);

    paint.setColor(Color.RED);

    if(line < credits.size()){
        paint.setTypeface(res.getFont());
        paint.setTextSize(fontSize);

        for (int i = 0; i < credits.get(line).length; i++)
            c.drawText(credits.get(line)[i], WIDTH / 2, y + i * fontSize, paint);
    }else{
        paint.setTypeface(res.getTitleFont());
        paint.setTextSize(240f);
        c.drawText("The End", WIDTH / 2, HEIGHT / 2, paint);
    }
}
```

Kuva 17. CreditScenen piirtometodi

5.2.5 Materiaalien käyttöönotto

Kun pelin mottori oli valmiiksi koodattu, voitiin aloittaa materiaalien käyttöönotto. Materiaaleja on erilaisia ja niille täytyy koodata omat luokat. Käytännössä materiaaleja ovat kuvat, musiikit ja äänet. Kaikki materiaalit ladataan ResourceHandler nimisellä luokalla. Luokkaan on koodattu erilaiset metodit eri materiaalityyppien käyttöönottamiseen.

Kuvat ovat olennainen osa peliä. Niillä määritellään miltä mikäkin näyttää. Laatikoiden sijaan pelaaja ja viholliset näyttävät joltakin. GameObject luokassa on muuttuja, johon kuva säilötään. Jokaiselle objektille ladataan kuva kun luokka alustetaan. ResourceHandler lataa kuvan objektille kuvan hakumetodia käyttäen, kuten kuvasta 18 käy ilmi.

```
public Bitmap fetchImage(String name, String url){
    Bitmap img = null;
    try {
        img = BitmapFactory.decodeStream(getFile(url + "/" + name + ".png"));
    }catch (IOException e) {
        try {
            e.printStackTrace();
            img = BitmapFactory.decodeStream(getFile("debug.png"));
        }catch(IOException e1){
            e1.printStackTrace();
        }

    }
    return img;
}
```

Kuva 18. Kuvan hakumetodi

Jotkin kuvat eivät ole staattisia, vaan ne liikkuvat. Esimerkiksi päähahmon pitää kävellä ruudulla, ei liukua. Tätä varten luotiin oma Animation luokka, joka animoi kuvan kyseiselle luokalle. Jotta animaatioluokka voi animoida kuvan, täytyy sille antaa taulukkokuva, jossa on jokainen tarvittava animaation osa. Kävelyanimaatiossa on erillinen kuva jokaisesta kävelynvaiheesta. Kuvasta 19 näkyy animaation rakennusmetodi, joka käsittelee kaikki taulukon kuvat. Kun taulukko käsitellään, syntyy animaatio. Animaatioluokka palauttaa aina yhden kuvan taulukosta kerrallaan ja menee yhden kuvan eteenpäin. Tällöin näyttää siltä, kuin hahmo kävelisi.

```
public Animation(Bitmap sheet, int frames, long delay){
    this.frames = new Bitmap[frames];
    this.delay = delay;

    int width = sheet.getWidth() / frames;
    int height = sheet.getHeight();

    for (int i = 0; i < frames; i++){
        this.frames[i] = Bitmap.createBitmap(sheet, width * i, 0, width, height);
    }

    currentFrame = 0;
    startTime = System.nanoTime();
}
```

Kuva 19. Animaatioluokan rakennusmetodi

Musiikkia varten luotiin AudioPlayer luokka, jolla toistetaan musiikkia. Kuvan tavoin, ResourceLoader hakee musiikin AudioPlayer luokalle. AudioPlayer on paneelin staattinen muuttuja ja sille on annettu staattinen metodi, jolla toistetaan musiikkia. Metodille annetaan musiikkitiedoston nimi. Metodi pysäyttää nykyisen musiikkinsa ja pyytää ResourceLoaderilta uutta musiikkia.

```
//Constructor
public AudioPlayer(String name, Context context, boolean loop) {
    fileName = name;
    this.context = context;
    playAudio(loop);
}
```

Kuva 20. AudioPlayerin rakennusmetodi

Äänet käyttävät samaa luokkaa kuin musiikki, mutta äänet eivät toistu useasti, toisin kuin musiikki. Äänille luotiin oma staattinen metodi, jotta olisi helppoa kutsua ääniä. Samoin se myös helpottaa löytämään kaikki koodit, joissa halutaan toistaa ääniä.

```
public static void playSound(String name){
    if(sfx != null)
        sfx.release();

    sfx = new AudioPlayer("sfx/" + name + ".ogg", context, false);
}
```

Kuva 21. Äänten toistamiseen käytettävä staattinen metodi

6 Testaus

Peliä testattiin jatkuvasti kehittäjän testaamana. Samalla peliä annettiin käyttäjille testattavaksi palautteen toivossa.

6.1 Debuggaus

Kehittäjä testasi ja debuggasi peliä jatkuvasti kehityksen aikana kahdella eri laitteella: Samsung Galaxy S 4 puhelimella, jossa oli Android versio 5.0.1, ja Lenovon S60000L-F tabletilla, jossa oli Android versio 4.2.2. Pääsääntöisesti testaus tapahtui Lenovon tabletilla. Pelin jokainen osio testattiin suoraan tabletissa, jotta saataisiin mahdollisimman aito kuva pelin toiminnasta.

Debuggauksen aikana ilmeni selkeä syy testata peliä kahdella eri laitteella. Jotkin osiot pelissä toimivat eri tavalla puhelimessa ja tabletissa. Tämä johti ratkaisujen uudelleen miettimiseen. Esimerkiksi osa fonteista ei toiminut puhelimella, vaikka tabletilla kaikki näytti toimivan hyvin.

6.2 Käyttäjätestaus

Käyttäjätestausta tehtiin projektin kehityksen aikana. Peliä annettiin käyttäjille testattavaksi ja heiltä kerättiin palautetta kysymällä heidän mielipiteitä valinnoista kuten käyttöliittymän sijainnista ja näkyvyydestä. Samalla käyttäjät antoivat palautetta omien näkemyksien mukaan eri ominaisuuksista kuten ohjattavuudesta.

Muutoksia tehtiin käyttäjätestauksen perusteella. Tappelussa nappuloille tehtiin erillinen palkki käyttäjätestauksen perusteella, jonka avulla estetään pelaajan hyppääminen, jos pelaaja ei osukkaan hyökkäysnappiin. Myös vaikeusaste nousi kun vihollisen hyökkäysmittari poistettiin käyttäjätestauksen perusteella.

Käyttäjätestausta tehtiin samoilla alustoilla kuin debuggaus eli Samsung Galaxy S 4 puhelimella ja Lenovon S60000L-F tabletilla.

7 Julkaisu Google Playssa

Peli on tarkoitus julkaista Googlen Play kaupassa. Jakamiskanavaksi valittiin Google Play, koska se on suurin ja virallisin Android kauppa, josta saa Androidille sovelluksia.

Peli on tarkoitus julkaista ilmaisjakeluna, koska tämä projekti ei ole tarkoitettu kaupalliseen käyttöön.

7.1 Projektin rakentaminen apk muotoon

Projekti rakennetaan ja pakataan apk (Android Application Package) muotoon, jonka pelaajat voivat sitten ladata Play kaupasta. Apk paketin avatessa peli asentuu käyttäjän Android käyttöjärjestelmään. Julkaisuversioon tarvitaan allekirjoitus apk:ta varten, jotta voidaan tunnistaa pelin tekijä. Ilman allekirjoitusta peliä ei voida tuoda apk muotoon. Android Studiossa on "Generate Signed APK" kohta, josta voi luoda itselleen allekirjoituksen. Aluksi pitää luoda uusi avainvarasto. Avainvarastoon pitää antaa seuraavat tiedot: Tiedoston nimi ja sijainti ja salasana. Samalla voidaan luoda uusi avain. Avaimelle pitää antaa seuraavat tiedot: Alias, salasana, voimassaolo aika, etu-ja sukunimi, organisaatio (jos yritys projekti) kaupunki, lääni ja maakoodi.

Kun on tiedot voimassa, voidaan kytkeä allekirjoitus projektiin. Android Studiossa sen voi laittaa File ->Project Structure -> Signing. Käytetään äsken luomaa avainvarastoa ja allekirjoitusta. Sen jälkeen liitetään se julkaisu build typeen. Kun apk tiedossa on allekirjoitus, voidaan se viedä Google Play kauppaan.

7.2 Google Play kauppaan vienti

Ensimmäisenä on luotava Google tili, jotta voidaan käyttää Google Play palvelua. Tilin luominen on maksuton. Kun tili on luotu, pitää se rekisteröidä julkaisijaksi. Tilin rekisteröimisestä veloitetaan 25 $ (23,52 €).

Tämän jälkeen ollaan valmiita aloittamaan julkaisu prosessi. Julkaisu koostuu vähintään neljästä eri osasta: apk tiedoston viennistä, kauppasivun valmistelusta, ikäluokituksesta, hinnoittelusta ja jakamisesta.

Apk tiedoston voi julkaista kolmessa eri tilassa: tuote, beta tai alpha. Koska tämä projekti julkaistaan valmiina, ei käsitellä betaa eikä alphaa. Kun tuote on ladattu, näkyy se ruudulla. Versiota voi päivittää tuomalla uuden apk tiedoston.

Kaupan sivulle annetaan projektin nimi, lyhyt kuvaus ja laajempi kuvaus. Kauppaan voi tuoda kuvia ja niitä voi kohdistaa eri laitteille. Esimerkiksi pieniä kuvia voi kohdistaa älypuhelimille. Pakollisia kuvia ovat isokokoinen kuvake (512x512) ja taustakuva (1024x500). Kauppaan voi myös laittaa videon. Kauppaan pitää myös määritellä projektin tyyppi (sovellus vai peli), kategoria ja ikärajoitus. Lopuksi kauppaan pitää laittaa sähköpostiosoite mahdollista yhteydenottoa varten. Osoite näkyy julkisesti kaupan sivulla.

Ikäluokitus koostuu lähinnä kysymyksistä, jotka luovat pelille ikäluokituksen. Kyselyssä kysytään mm. väkivallasta, pelosta, seksuaalisuudesta, uhkapelaamisesta, kielenkäytöstä, huumeiden käytöstä ja muuta yleistä ikäluokitukseen kuuluvaa. Kun kyselyyn on vastattu, luo se projektille automaattisesti ikäluokituksen eri puolille maailmaa. Tämä projekti sai ikäluokituksen kaikille sopivaksi.

Tuotteelle voi myös määrittää hinnan. Hintaa varten täytyy luoda oma kauppiastili. Koska tämä projekti on ilmainen, ei tätä prosessia käydä läpi. Tuotteelle pitää määrittää jakelumaat. Voit esimerkiksi jakaa tuotettasi pelkästään Suomessa. Projektin voi määrittää eri Android laitteille. Tässä listassa niitä voi kategorisoida Andoid Wearille, Android TV:lle tai Android Autoon. Tämä projekti on suunnattu mobiililaitteille (puhelin ja tabletti), joten näitä rajoituksia ei valita. Projektin voi myös määrittää, onko se suunnattu perheille, työhön tai koulutukseen. Tämä projekti ei ole mihinkään näistä suunnattu.

Kun kaikki vaiheet on käsitelty, voidaan tuote julkaista kauppaan. Sovellus tarkistetaan ennen kuin se laitetaan julkiseen jakoon. Tässä voi kestää useita tunteja.

8 Tulokset

Projektin tuloksena syntyi julkaisuvalmis peli Android laitteelle, joka pystytään pelaamaan alusta loppuun. Projekti myös saatiin julkaistua Google Play kauppaan, mikä oli toivottua. Projektiin on myös helppo kehittää lisää tasoja, vihollisia ja iskuja, koska nämä osiot eivät ole kovakoodattu peliin.

Projektin aikana selvisi, että Java koodaus Androidille on melkeinpä samanlaista kuin tietokoneelle. Osa kirjastoista oli valmiina Androidille, joita ei voida käyttää tietokoneella ja päinvastoin tietokoneella on osa kirjastoista, joita ei voida käyttää Androidilla. Muutoin koodaaminen oli melkeinpä samanlaista.

Projektin käynnistys Androidilla ei tuottanut ongelmia. Android Studiolla voitiin ajaa peliä suoraan Android laitteella, jos se oli kytkettynä koneeseen. Pelin pystyi myös lataamaan laitteelle apk muodossa, josta se sitten asennettiin laitteelle kokeilua varten. Kauppaan vietävää apk tiedostoa varten piti luoda allekirjoitus, koska Google Play kauppaan ei voi viedä allekirjoittamattomia sovelluksia. Allekirjoituksen pystyi luomaan suoraan Android Studiolla.

9 Pohdintaa

Projekti tuli tehtyä hiukan kiireellä. Projekti saatiin julkaistua Googe Play kauppaan aikataulussa, mutta osa ominaisuuksista jäi kokonaan pois. Esimerkiksi taustakuvan piirto olisi tuonut peliin lisää eloa, mutta ajanpuutteen takia ideasta jouduttiin luopumaan. Syynä ei ollut niinkään taustakuvien puute, vaan optimisaatiossa esiintynyt pulma. Kuvaa ei saatu piirtämään ilman, että suorituskyky olisi pysynyt samana. Kuvan piirto tiputti suorituskykyä huomattavasti. Samalla tuli kiire tuottaa iskuille kuvat ja animaatiot. Ajanpuutteen vuoksi, käytettiin usein samaa kuvaa.

Projekti oli selvästi liian iso näin pienelle aikataululle. Vaikka projekti vaikutti pieneltä opinnäytetyön alussa, huomattiin projektin tekemisen aikana, että kyseessä onkin isompi projekti. Itse peli on hyvin yksinkertainen ja lyhyt mutta jopa sen tekemiseen meni paljon aikaa. Varsinkin kun materiaalia piti saada jokaiselle viholliselle ja jokaiselle iskulle. Tiimissä tämä olisi voinut olla ihan hyvän kokoinen projekti mutta yksintekevälle se oli liian suuri.

Projektissa myös ilmenee bugeja, joita ei ehditty korjata. Esimerkiksi peliä ei voi jatkaa kun vaihtaa ruutua esimerkiksi vastatakseen puhelimeen. Ajanpuutteen vuoksi, näitä ei ehditty korjata.

10 Yhteenveto

Projektin tavoitteena oli tuottaa Project Katz -peli Android käyttöjärjestelmälle ja Julkaista se Google Play kauppaan.

Projektin toteuttamiseen tehtiin suunnitelma, millainen peli ja mitä siinä tehdään. Projektin suunnitelmissa selvisi kaikki tasot, viholliset ja iskut. Projekti toteutettiin Java ohjelmointikielellä. Projektin organisoinnin vuoksi projekti jaettiin eri osa-alueisiin. Tämä helpotti luokkien löytöä ja ylläpitoa.

Google Play kaupassa julkaisemiseen tarvittiin Google tili ja se piti rekisteröidä julkaisijaksi. Tämän jälkeen saatiin tuotua projekti Google Play kauppaan.

Oppimiskokemuksena tämä oli hyvin rikas kokemus. Alkuperäistä kokemusta Android järjestelmälle ohjelmoinnista ei ollut eikä myöskään Android Studion käytöstä. Nämä olivat suuria kokonaisuuksia pienellä aikataululla, mutta Javan olessa tuttua, tuli tutumpi olo jo heti alkukankeuden jälkeen.

Pelien suunnittelu on suuri osa pelien alkua. Ilman suunnitelmaa ei synny peliä. Suunnitelma pitää projektin kasassa ja estää sen liiallisen paisumisen. Helposti tulee mieleen ominaisuuksia, joita haluaisi lisätä kesken toteutuksen, joka taas pidentää pelin koodaamisaikaa.

Pelien koodaamisessa tuli hyvää oppimista. Vaikka pelit ovat joskus yksinkertaisia, on niiden koodaminen aivan toista. Pelilogiikan tajuaminen kehittyy aina uuden pelin luodessa. Tekemällä oppii myös eri menetelmiä eri osa-alueille. Esimerkiksi ulkoisten resurssien käyttö kuten XML-tiedostot helpottivat pelin tekemistä.

Lähteet

Cartesian. The Rise of Mobile Phones: 20 Years of Global Adoption. Luettavissa: http://www.cartesian.com/the-rise-of-mobile-phones-20-years-of-global-adoption/. Luettu: 8.11.2015

Cutlack, G 2012. What is Google Play?. Luettavissa: http://www.techradar.com/news/phone-and-communications/mobile-phones/what-is-google-play-1073348. Luettu: 8.11.2015

Gallardo, R, Hommel, S, Kannan, S , Gordon, J & Zakhour, S 2015. The Java Tutorial: A Short Course on the Basics. Oracle.

Gargenta, M 2011. Learning Android, s. 1-5. O'Reilly Media Oy.

Google a. Android Studio Overview. Luettavissa: https://developer.android.com/tools/studio/index.html. Luettu 8.11.2015

Google b. Android, the world's most popular mobile platform. Luettavissa: https://developer.android.com/about/android.html. Luettu: 8.11.2015

IDC. Smartphone OS Market Share, 2015 Q2 Luettavissa: https://www.idc.com/prodserv/smartphone-os-market-share.jsp. Luettu: 8.11.2015

Open Game Art. FAQ. Luettavissa: http://opengameart.org/content/faq. Luettu: 8.11.2015

Oracle. Learning About Java Technology. Luettavissa: https://java.com/en/about/. Luettu: 8.11.2015

Protalinski, E. 2014. Google releases Android Studio 1.0, the first stable version of its IDE. Luettavissa: http://venturebeat.com/2014/12/08/google-releases-android-studio-1-0-the-first-stable-version-of-its-ide/. Luettu: 8.11.2015

Sharma, M & Thakur, A 2015. Review Paper on Android Operating System. Luettavissa: http://ijetst.in/ems/index.php/ijetst/article/view/703/568. Luettu: 8.11.2015