OULUN AMMATTIKORKEAKOULU

Tuomo Lauri

LIPPOAMISPELI ANDROID-PUHELIMIIN

LIPPOAMISPELI ANDROID-PUHELIMIIN

Tuomo Lauri
Opinnäytetyö
Kevät 2017
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikka, ohjelmistosuunnittelu

Tekijä: Tuomo Lauri
Opinnäytetyön nimi: Lippoamispeli Android-puhelimiin
Työn ohjaaja: Eino Niemi
Työn valmistumislukukausi ja -vuosi: kevät 2017
Sivumäärä: 20

Opinnäytteen tavoitteena oli luoda lippoamispeli Android-älypuhelimiin. Lippoaminen on kalastamiskeino suurella haavin kaltaisella pyydystämisvälineellä. Opinnäytetyö oli haastava ja mielenkiintoinen, sillä älypuhelimien pelit ovat kehittyneet pitkälle Nokian matopeleistä. Tavoitteena oli saada peliin muun muassa puhelimen liikeanturit ja värinä käyttöön, sekä grafiikkaa Kukkolankoskelta. Työn tilaaja oli Kukkolan kyläyhdistys.

Java-ohjelmointi oli entuudestaan vahvin koodauskielistäni. Ohjelmointiympäristönä toimi Android Studio. Pelin kehityksessä käytettiin Scrum-menetelmää. Vaatimuslistan muodostamisen jälkeen siirryttiin suunnitteluun. Toteutusvaiheessa tuotettiin testattava pelin versio aina, kun lisättiin uusia ominaisuuksia.

Ensimmäiset lippoamispelin versiot keskittyivät puhelimen ominaisuuksien käyttöönottoon. Kalojen lisäys muunsi teknologiademon peliksi. Viime silauksina tulivat äänet ja parempaa grafiikkaa. Kaikkia tavoitteita ja vaatimuksia ei saavutettu, mutta pelissä oli myös ominaisuuksia vaatimuslistan ulkopuolelta. Opinnäytetyön aikana oppi paljon itsenäisestä projektityöstä.

Asiasanat: Android, lippoaminen, pelit, älypuhelimet

SISÄLLYS

SANASTO

Dialogi	Android-ohjelmoinnissa käytettävä luokka, jonka tarkoituksena on näyttää muokattava ikkuna puhelimen näytöllä, joka voi sisältää tekstiä, kuvia ja painikkeita.
Eclipse	Avoimen lähdekoodin ohjelmointiympäristö. Tukee muun muassa Java, C++ ja PHP -ohjelmointikieliä.
Toast	Android-ohjelmoinnissa käytettävä luokka, jonka tarkoituksena on näyttää teksti-ikkuna puhelimen näytöllä.
Xamarin	Ohjelmointiympäristö, joka tukee muun muassa Android-ohjelmointia.

1 JOHDANTO

Älypuhelimet ovat nykyaikaa. Älypuhelin on laite, jolla voi vastaanottaa ja soittaa puheluita kuten tavallisella kännykällä, ja sen lisäksi se toimii kämmentietokoneena. Tavallisilla kuluttajilla on yhä enenevässä määrin mukana kulkeva laite, jolla he voivat suorittaa mitä monimutkaisimpia sovelluksia. Vuonna 2015 69 % suomalaisista käytti älypuhelinta. (1.)

Älypuhelimissa yleisin käyttöjärjestelmä on Android. Vuonna 2015 82 % maailman älypuhelimista käytti Android-pohjaista käyttöjärjestelmää. Android on vapaan lähdekoodin käyttöjärjestelmä, jossa on mahdollista ajaa kolmannen osapuolen tuottamia sovelluksia. (2.)

Kaikki älypuhelimiin suunnatut sovellukset eivät liity suoraan kommunikointiin tai työn tekemiseen, vaan osa on aivan puhtaasti viihdettä varten. Maailmankuulu Rovio näytti jo vuonna 2009 Angry birds -pelillään, että kännykkäpeleille on kysyntää. (3, s. 131) Tässä opinnäytetyössä on kyse lippoamispelistä Android-älypuhelimiin.

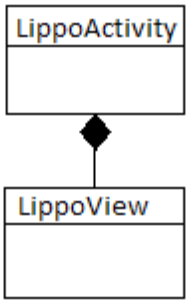
Opinnäytetyön teettäjänä on Kukkolakosken kyläyhdistys. Heidän toivomuksistaan laadimme yhteistyössä vaatimuslistan lippoamispelille. Pelin taustoina tuli olla kuvaa, videota tai animaatiota Kukkolankoskelta. Matka lippouspaikkaan olisi oma osionsa pelissä. Pelin tuli hyödyntää puhelimen liikeantureita niin, että peliä voi ohjata puhelinta liikuttamalla. Siian ja lohen lippoamiseen olisi oltava erilliset lippopaikat tai versiot. Peliltä vaadittiin 3D-grafiikkaa. Lippoamispelin tuli käyttää puhelimen värinää välittämään tietoa siitä, mitä lipolle tapahtuu. Lippopelin vedenpohjan tuli olla vaihteleva eri etäisyyksillä. Pelissä pisteitä annettiin pyydystettyjen kalojen painon mukaan.

Lippoaminen on kalastamismuoto, jossa kaloja pyydystetään suurella haavilla. Lipossa on pitkä, noin kuusimetrinen, varsi ja vanne, joka on vajaan metrin halkaisijaltaan. Vanne on varren päässä, joka taas on kiinni verkkopussissa eli havaksessa, johon kala tai kalat on tarkoitus pyydystää. Kukkolankoskelta lipotaan pääasiallisesti siikaa ja lohta.

2 SUUNNITTELU

Aloitin lippoamispelin suunnittelun heti aiheen saatuani. Päätin toimia Scrum-menetelmän mukaisesti, eli tekemällä testattavia ohjelmaversioita, joihin joka syklissä lisätään jotain uutta. Aluksi suunnittelin käyttäväni ohjelmointiympäristönä Eclipseä Android Suiten kanssa, mutta jo värinää toteuttaessa tuli selväksi, että olisi parempiakin vaihtoehtoja. Harkitsin Xamarinin asennusta, mutta päädyin kuitenkin Android Studio -ohjelmointiympäristöön. Tein valinnan sen perusteella, että olen ohjelmoinut Android Studiolla aikaisemminkin. Xamarinin tiesin vain nimeltä, enkä ollut varma, millaiset maksut tai oikeudet siihen vaaditaan.

Halusin erotella puhelimen suorituskykyä koettelevat toiminnot erilliseen luokkaan, joka on oma säikeensä erillään käyttöliittymästä. Pääluokan säikeessä taas reagoidaan käyttäjän toimiin. Pääluokka perii AppCompactActivity-luokan. Alaluokka perii SurfaceView-luokan joutuisaa piirtoa varten, ja siinä suoritetaan muutkin kuormittavat toimet. (Kuva 1.) Puhelimessa näkyvä grafiikka siis piirretään alaluokan LippoView säikeessä. Äänet lukeutuvat myös kuormittaviin toimenpiteisiin, joten nekin soitetaan alaluokan säikeessä. Liikeanturit hoidetaan pääluokan säikeessä. Ne selvittävät missä asennossa puhelin kulloinkin on.



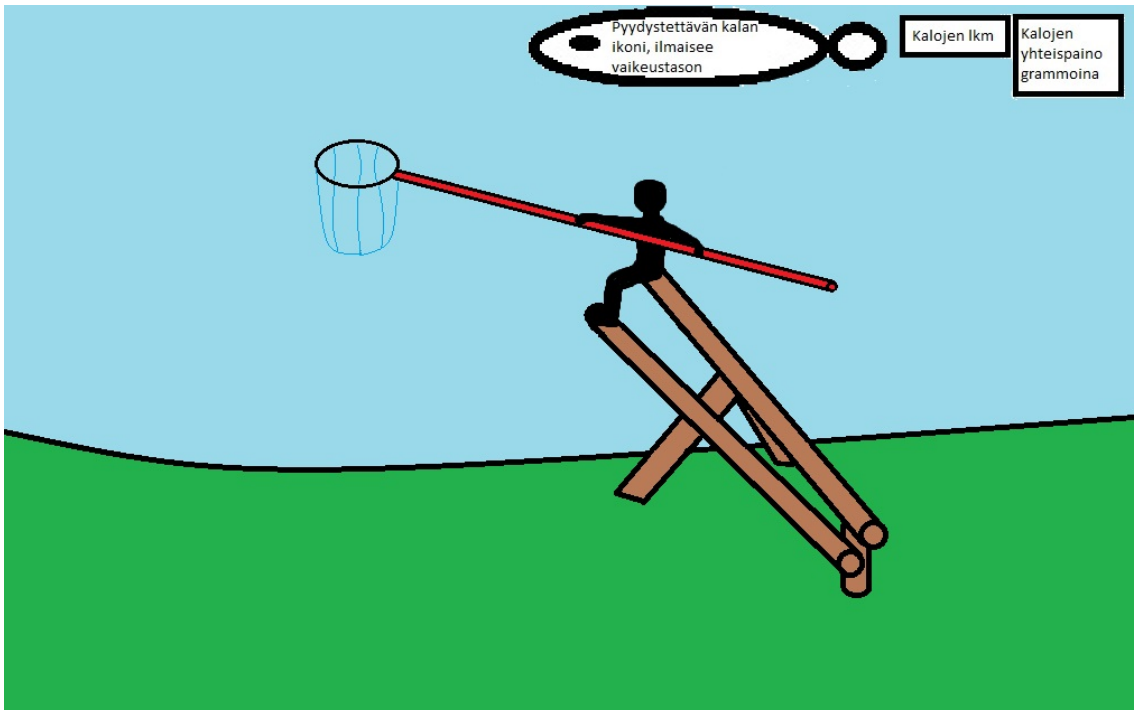
KUVA 1. Luokkakaavio

2.1 Vaatimuslista ja vaiheistus

Aluksi suunnittelin tekeväni version, jossa puhelimen kiihtyvyysanturien muutokset ilmenevät sovelluksessa jollain tavalla. Seuraavaksi halusin lisätä sovellukseen puhelimen värinän hallitsevan osion. Kolmas vaihe olisi yhdistää kaksi ensimmäistä ominaisuutta niin, että puhelin värisee sen mukaan, miten sitä liikutetaan. Seuraavaksi sovelluksen tuli liikuttaa esinettä tilan sisällä puhelinta kallistellessa ja väristä, jos esine osuu tilan reunoihin. Vaihe viisi olisi lisätä tilaan toinen esine. Ensimmäisellä esineellä törmätessään puhelin reagoisi värinällä. Seuraava versio väristäisi puhelinta eri tavalla sen mukaan, mihin puhelimen liikkeiden ohjaama esine törmää. Seitsemäs vaihe olisi lisätä puhelimen ohjaaman esineen, eli lipon nosto vedestä. Seuraavaan versioon lisättäisiin grafiikkaa. Lippo piirrettäisiin paksuna viivana, joka kääntyisi akselin ympäri tiettyihin raja-arvoihin. Kalat olisivat elementtejä, jotka kulkevat satunnaiseen suuntaan, kunnes ne törmäisivät johonkin, jolloin ne vaihtaisivat suuntaa. Viimeinen vaihe olisi lisätä matka lippouspaikkaan, jossa ideana on liikkua tuetuilla lankuilla tai kuivilla kivillä, välttäen tukemattomia lankunpäitä ja märkiä kiviä.

2.2 Ruutusuunnitelma

Aluksi tein ruutusuunnitelman, joka kuvasti, miltä lippouspeli näyttää valmiina ja mitä tietosisältöä siinä esitetään (kuva 2). Pelissä tulisi näyttää pyydystettävän kalan ikoni, kalojen lukumäärä ja kalojen yhteispaino grammoina. Pyydystettävän kalan ikoni ilmoittaisi vaikeustason. Suunnittelin erilaiset vaikeudet sen perusteella, pelataanko siikalipolla vai lohilipolla. Myöhemmin huomasin, että erilliset lipot ovat vain lainsäädännöllisistä syistä. Pelissä ei tarvitse murehtia lain rikkomisesta, joten pelissä ei tarvitse vaihtaa lippoa. Lipon tulisi pysytellä keskellä ruutua, niin että pelaajalla on jatkuvasti tiedossa, mikä on tilanne. Veden osuus taustasta olisi myös suurempi kuin maan, sillä maan merkitys pelin kannalta on pienempi. Myöhemmissä versioissa vähensin entisestään maan osuutta, jotta suurempi osa älypuhelimen näyttöpinta-alasta olisi pelattavaa aluetta.



KUVA 2. Ruutusuunnitelma

3 TOTEUTUS

Toteutusvaihe on se, johon kului aikaa eniten. Olen koodannut Java ME:llä yhden projektin ja käynyt monta Java-kurssia, joten Java oli tuttu ohjelmointikieli. Android-ohjelmointia käytiin läpi parissa kurssissa myös, joten alkuasetelma oli hyvä ennen opinnäytetyön aloitusta. Milloinkaan aikaisemmin en ollut kuitenkaan luonut tyhjältä pohjalta näin suurta kokonaisuutta.

3.1 Luokkasokkelo

Suurimmaksi esteeksi nopeasti nousi esiin se, että useimmat esimerkit ja ohjeet neuvovat perimään jonkin luokan. Kukin luokka voi Androidissa periä vain yhden luokan. Niinpä jos ei halua tehdä monta luokkaa, jotka sisältävät pieniä pätkiä koodia, on perehdyttävä syvemmin jokaiseen uuteen lisättävään ominaisuuteen, ja miten se onnistuu ilman perintää.

3.2 Uudelleenaloitus

Ensimmäinen saavutukseni oli, kun sain kiihtyvyysanturin toimimaan ja puhelimen värisemään sen mukaan. Ensimmäiset kolme vaihetta täytyi kuitenkin aloittaa alusta, sillä ensimmäisellä kerralla en huomannut, että Activity-luokan näkymän käyttö ei ole pakollista. Lopullisessakin toteutuksessa pääluokka perii Activity-pohjaisen luokan, mutta näytöllä näytetään näkymä, joka piirretään alaluokassa. Uudelleenaloitus puhtaalta pöydältä oli silti hyödyllistä. Sain karsittua pois ylimääräisen luokan, joka hoiti ainoastaan puhelimen kiihtyvyysanturin. Sitä ei tarvita lopullisessa pelissä, ainoastaan testaamisessa.

3.3 Piirto

Uudelleenaloituksen jälkeen sain nopeasti toimivan värinän ja onnistuin näyttämään pelialustan, johon pystyin piirtämään. Perusideana on, että lippoaja ei liiku, eli akseli, jonka ympäri pyöritetään lippoa, on aina kiinteä. Aluksi piirsin taustakuvan ja siihen päälle liponvarren, jota käänsin pyörittämällä koko pelialustaa. Huomasin pian kuitenkin ongelman, jos pyöritän koko pelialustaa, akselit taustakuvan ja lipon kannalta ovat erilaiset.

Minun siis täytyi muuttaa lipon liikuttamistapaa. Aikaa kului uuden pyöritysmekanismin ohjelmoinnissa, mutta samalla huomasin, että vaikka jotain näyttävää saa nopeasti ja helposti aikaan, se ei aina ole paras ratkaisu. Korjatun pyöritysmekanismin ansiosta taustakuvan ja lipon koordinaatisto olivat samat, joten törmäystarkistus olisi mahdollista.

Lisäsin piirrossa lipon varteen lipon ringin ja lippoverkon. Lippo vielä tässä vaiheessa kääntyi nopeasti ja lipon asento vastasi aina puhelimen asentoa, eli puhelinta joutui pyörittelemään käsissä. Kuvan 3 koodissa piirretään lipon varsi. Siinä toivossa, että suorituskyky paranisi, tallensin ensin usein käytettävät välitulokset väliaikaisiin muuttujiin. Lopullinen piirto riippuu siitä, onko lipon varsi vedessä vai ei. Vedessä oleva osio varresta piirretään eri värillä. Kalojen piirto hoitui aluksi pyöreillä palloilla. Kalan koko riippui siitä, miten painava kala oli kyseessä. Myöhemmin vaihdoin kalat palloista kalan kuviksi.

```
// lipon varren piirto
mLinePaint.setStyle(Paint.Style.STROKE) ;
mLinePaint.setStrokeWidth(10) ;
mLinePaint.setColor(Color.RED) ;
float cosLippoangle = (float) Math.cos((double) lippoAngle / -57) ;
float sinLippoangle = (float) Math.sin((double) lippoAngle / -57) ;
float lippoEndX = lippoVerkkoCoordinates[0] + (cosLippoangle * (lippoDiameter / 2)) ;
float lippoEndY = lippoVerkkoCoordinates[1] + (sinLippoangle * (lippoDiameter / 2)) ;
float lippoBeginX = lippoajaCoordinates[0] + (cosLippoangle * (lippoLength / 2)) ;
float lippoBeginY = lippoajaCoordinates[1] + (sinLippoangle * (lippoLength / 2)) ;
if(lippoAngle > 0 || lippoAngle < -180) {
    canvas.drawLine(lippoEndX , lippoEndY, lippoBeginX, lippoBeginY, mLinePaint) ;
}
else {
    canvas.drawLine(lippoEndX + (lippoLength / 3 * cosLippoangle) * sinLippoangle,
            lippoEndY + (lippoLength / 3 * sinLippoangle) * sinLippoangle, lippoBeginX, lippoBeginY, mLinePaint) ;
    mLinePaint.setColor(Color.BLUE) ;
    canvas.drawLine(lippoEndX, lippoEndY, lippoEndX + (lippoLength / 3 * cosLippoangle) * sinLippoangle,
            lippoEndY + (lippoLength / 3 * sinLippoangle)* sinLippoangle, mLinePaint) ;
}
```

KUVA 3. Koodi lipon varren piirrosta

3.4 Kalat

Lisättäessä kaloja pelialueelle aluksi ne sijoittuivat sinne tänne, jotkin maalle ja jotkin nurkkiin, jonne lipolla ei ylety, eivätkä kalat liikkuneet. Lipon osuessa kalaan puhelin värisi pienillä tauoilla, kunnes lipon käänsi maalle päin. Kun kalan sai maalle, peli ilmoitti kalalajin ja kalan painon Toast-viestillä ja peli meni tauolle.

Kalan pyydystys laskettiin onnistuneeksi, kun kala oli lipossa ja lippo käännettynä jommallekummalle puolelle ääriasentoon. Vielä tässä vaiheessa kalojen painot arvottiin sattumanvaraisesti ja ne saattoivat olla epärealistisia.

3.5 Kalojen liikuttaminen

Suuri saavutus oli, kun sain kalat liikkumaan. Tässä vaiheessa pystyi jo pelaamaan lippoamispeliä, vaikka selkeitä puutteita vielä löytyikin. Enää ei haitannut, jos kala sattui syntymään lipon ulottumattomiin, sillä ennen pitkää se uisi pyydystysetäisyydelle. Sain myös korjattua ongelman, jossa kaikki kalat arpoutuivat vasempaan ylänurkkaan. Samalla lisäsin pyydystettävien kalojen joukkoon kiviä. Kalan tai kiven rantaan raahatessa tuleva toast viesti korvautui dialogilla.

3.6 Orientaation muutokset

Orientaation muuttaminen tuotti minulle runsaasti päänvaivaa. Jouduin miettimään, mitä kaikkea tarvitsen siihen, että pelitilanne pysyy samana. Kaikki tarvittavat muuttujat täytyi tallentaa ja ladata uudestaan jokaisen orientaatiomuutoksen jälkeen. Myös dialogi täytyi säilyttää orientaation muutoksessa. Ongelmatilanne, mihin törmäsin oli se, että peli jatkui paussilta ja dialogi hävisi, kun orientaatio muuttui, vaikka peli aiemmin oli asetettu paussille.

Muutin lipon pituuden riippuvaiseksi näytön koosta, ettei kävisi niin, että suuren resoluution älypuhelimilla lipon varsi olisi hyvin lyhyt pyydystysalueeseen nähden. Tämä muutos auttoi myös lipon varren pituuden suhteutusta pelialueeseen puhelimen vaaka-asennossa. Vähensin myös maan osuutta pelialueesta, sillä sen pelillinen merkitys oli hyvin vähäinen. Kuvan 4 koodissa muutetaan pelissä käytettävää koordinaatistoa, mikäli se on tarpeellista, sen mukaan, missä asennossa puhelin on. Oletuksena on, että puhelimen näyttö on pystyasennossa. Tietääkseni nykyiset puhelimet eivät käännä näyttöä ylösalaisin, mutta sekin vaihtoehto on koodissa otettu huomioon.

```
int screenRotation ;
screenRotation = mWindowManager.getDefaultDisplay().getRotation() ;

if (screenRotation == Surface.ROTATION_0) {
    X = SensorManager.AXIS_X ;
    Y = SensorManager.AXIS_Z ;
}
else if (screenRotation == Surface.ROTATION_90) {
    X = SensorManager.AXIS_Z ;
    Y = SensorManager.AXIS_MINUS_X ;
}
else if (screenRotation == Surface.ROTATION_180) {
    X = SensorManager.AXIS_MINUS_X ;
    Y = SensorManager.AXIS_MINUS_Z ;
}
else if (screenRotation == Surface.ROTATION_270) {
    X = SensorManager.AXIS_MINUS_Z ;
    Y = SensorManager.AXIS_X ;
}

float[] adjustedRotationMatrix = new float[9] ;
SensorManager.remapCoordinateSystem(rotationMatrix, X, Y, adjustedRotationMatrix) ;

float[] orientation = new float[3] ;
SensorManager.getOrientation(adjustedRotationMatrix, orientation) ;
```

KUVA 4. Koodi koordinaatiston asetuksesta

Pieni mutta merkittävä muutos oli, kun lisäsin peliin ominaisuuden, ettei näyttö himmene tai sammu nopeasti. Aiemmin peliä pelatessa näyttöön ei tarvitse juurikaan koskea, jos kalaa ei nouse jatkuvasti. Kun vain kääntelee puhelinta kaloja jahdatessa, puhelin luulee, että on sopiva aika säästää virtaa, ja sammuttaa näytön.

3.7 Äänet

Äänien lisäys toi peliin paljon. Tähän asti ainoa ääni oli lähtenyt puhelimen värinästä mekaanisista syistä. Äänien saaminen kuulumaan ei ollut erityisen ongelmallista. En ollut aikaisemmin käyttänyt Teosto-vapaita ääniä tai tutustunut niiden käyttöehtoihin tai siihen, mistä niitä ylipäätänsä löytää. Oikeuksien selvittely ja sopivien äänien löytäminen tekivät äänien lisäämisestä peliin aikaa vievää puuhaa. Ensimmäiseksi lisäsin kosken ääniä tuomaan tunnelmaa taustaäänenä. Myöhemmin ääniin kuului intromusiikki, kalan loiskimista lipossa, tuulettelu kalan saamisesta maalle ja ruudunläpäisymusiikki.

Äänien soittoa tarvittiin niin usein, että loin sille oman metodin (kuva 5). Metodi playSound vastaanottaa neljä muuttujaa. Ensimmäisellä muuttujalla valitaan numeraalisesti, mitä äänisoitinta käytetään. Jotta taustaääntä voidaan soittaa yhtäaikaisesti muiden äänien kanssa, yksi äänisoitin ei riitä. Päädyin käyttämään kahta soitinta, taustaäänien soittoon tarkoitettua soitinta ja pelissä aktiivisia tapahtumia säestävää soitinta. Metodin toinen muuttuja kertoo, mitä ääniresurssia soitin käyttää äänilähteenä. Kolmas muuttuja on volyymi. Neljäs muuttuja määrittää, soitetaanko äänilähde kerran läpi vai toistetaan aina alusta, kun se on soitettu loppuun.

```
public void playSound(int player, int mediaResource, float volume, boolean looping) {
    if(player == 1) {
        if (bgMediaPlayer == null) {
            bgMediaPlayer = MediaPlayer.create(this.getContext(), mediaResource) ;
            bgMediaPlayer.setVolume(volume, volume) ;
            bgMediaPlayer.setLooping(looping) ;
            bgMediaPlayer.start() ;
```

KUVA 5. Koodi äänensoittometodista

3.8 Pyydystys

Pelin kehitys oli edennyt jo niin pitkälle, että seurasi jo pelillinen muutos. Kaloja pystyi aikaisemmin saamaan kiinni pelkästään sillä, kun lipon rinki sattui osumaan tarpeeksi lähelle kalaa. Muutin peliä lähemmäs reaalimaailman lippoamista niin, että vain myötävirtaan lippoa liikuttaessa voi pyydystää mitään. Periaatteessa kiviä voisi pyydystää vastavirtaankin. Tein kuitenkin päätöksen, että on kyseessä fiksu lippoaja, joka nostaa lipon vedestä vastavirtaan lippoa siirtäessä, säästääkseen voimia. Tähän liittyen muutin peliä niin, että virta vie lippoa, kun lippo on vedessä. Tällä muutoksella kannustetaan pelaajaa lippoamaan pidemmillä vedoilla, sen sijaan että lippoa pitää paikallaan, kunnes kala sattuu kohdalle.

Tässä vaiheessa lisäsin peliin tulokset. Pisteet laskettiin pyydystettyjen kalojen yhteispainosta grammoina. Kivistä ei saanut pisteitä. Lipon hallintaa myös muutettiin radikaalisti. Aikaisemmin lippo seurasi suoraan puhelimen asentoa ja käänsi lippoa osoittamaan samaan suuntaan. Huomasin, että tuodessa kalaa

maalle puhelinta joutui kääntämään todella paljon ja pelin seuraaminen muuttui hankalaksi. Vaihdoin lipon ohjauksen suhteelliseksi aloituskulmaan, jota pystyi muuttamaan laittamalla pelin tauolle. Siis jos kääntää puhelinta alkuasennosta hieman vasemmalle, lippo kääntyy vasemmalle niin kauan, kunnes se päätyy ääriasentoon vasemmalle.

Katsoin, että valikoille ei ole tarvetta. Pelin ohjeet annetaan pelin käynnistyessä dialogissa, jonka pelaajan pitää hyväksyä pelin aloittaakseen. Ohjeet näytetään myös, kun peli asetetaan tauolle. Pelin voi laittaa tauolle koskettamalla näyttöä.

4 TESTAUS

Testaus suoritettiin mahdollisimman usein, kunhan vain koodi kääntyi ja testattava ominaisuus saatiin jotenkin ilmenemään ohjelmassa. Aivan aluksi testasin lippoamispeliä emulaattorilla, mutta lopullinen tapa, millä peliä pelataan, on älypuhelimella. Kaikkea ei voi testata emulaattoreilla tai testaus jää vajavaiseksi, esimerkiksi värinän ja liiketunnistuksen osalta. Siispä päätin siirtyä testaamaan pääasiallisesti omalla Android-puhelimellani Samsung Galaxy S2.

Puhelimeni on testaukseen oivallinen, sillä se on hyvin varhaisia Android-puhelimia. Sen Android-versio on enemmän rajoittava ja kaikki Androidit ovat taaksepäin yhteensopivia. Toinen etu vanhan älypuhelimen käytöstä on, että sen prosessorin tehot ovat vaatimattomat nykyisiin älypuhelimiin verrattuna. Jos peli toimii minun puhelimessani sulavasti, se toimii miltei kaikissa Android-puhelimissa riittävällä nopeudella.

4.1 Kiihtyvyysanturi ja värinä

Kiihtyvyysantureiden toimivuus oli helppo testata. Käynnistin ohjelman ja heiluttelin puhelinta rivakasti. Mikäli näytöllä olevat arvot muuttuivat lukemiltaan vain silloin, kun heilutin puhelinta, ja pysyivät samoissa lukemissa, kun puhelin oli paikallaan, testi oli läpäisty. Arvoja oli tosin hankala lukea puhelinta liikuttaessa, joten värinän lisäys helpotti kiihtyvyysanturien testausta. Asetin tietyn raja-arvon kiihtyvyydelle, jonka jälkeen puhelin värisisi. Sitten heiluttelin puhelinta enenevässä määrin. Mikäli puhelin värisi vain räväkällä heiluttelulla eikä vähäisellä heilautuksella, toimivuus oli testattu kymmenen testin jälkeen, kun ongelmia ei löytynyt.

4.2 Piirto ja törmäystarkistus

Piirron testaaminen oli pääsääntöisesti yksinkertaista. Jos jokin ei näkynyt, oli jotain pielessä. Törmäystarkistus oli ensimmäinen vaihe, jossa testaus ei helposti näyttänyt virheitä. Testasin törmäystarkistusta pyydystämällä pelissä kaloja, jotka piirrettiin palloina vielä tässä vaiheessa. Mikäli kala oli havaksen alueella, kun se tuli pyydystetyksi kymmenen kertaa ilman ongelmia, katsottiin törmäystarkistuksen toimivan. Pitkään käytin törmäystarkistukseen neliskanttisia alueita, vaikka pyydystettävät ja lipon haavi olivatkin pyöreitä. Pyydystäminen vain oli niin hankalaa, että sen tarkka tutkiminen oli hyvin haastavaa.

4.3 Säikeet

Säikeiden määrän testaus oli myös haastavaa. Vain pelin kaatumisongelmista pystyi arvailemaan, että jotain oli vialla. Tietyissä tilanteissa käynnistettiin useampia säikeitä, vaikka tarkoitus oli rajoittaa ne kahteen, yksi säie käyttöliittymälle ja toinen laskennalle ja piirrolle. Epävakauden testaamiseen en tiennyt erillistä testauskeinoa, joten muun testauksen ohessa pistin merkille, kuinka usein peli kaatui selittämättömästä syystä. Katsoin tämän ongelman ratkaistuksi, kun viikon testauksen aikana ei ollut tullut yhtään selittämätöntä kaatumista.

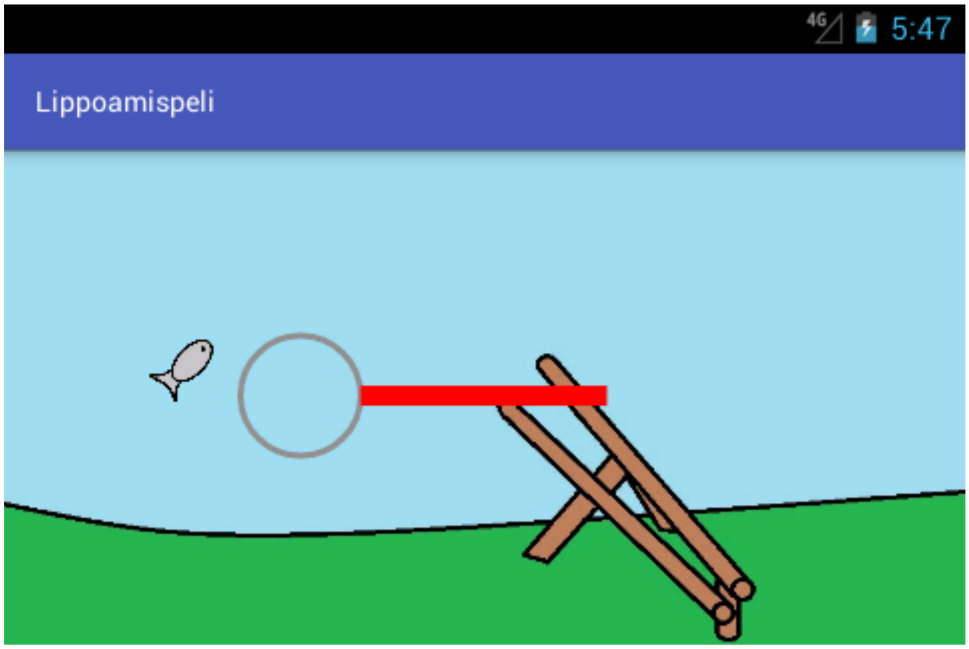
4.4 Orientaation muutos

Sitten löytyi selkeä puute. Olin huomannut, että kun vaihdoin puhelimen pystyasennosta vaaka-asentoon, vaikka peli oli paussilla aikaisemmin, se ei enää vaihdon jälkeen ollutkaan. Tarkemmin tutkiessa myös pyydystettäviä oli taas maksimimäärä ja niiden koko oli eri kuin aikaisemmalla orientaatiolla. Selvisi, että aina orientaatiota muuttaessa peli käynnistyi alusta. Testasin orientaation muutoksen vaikutusta pyydystämällä muutaman pyydystettävän ja pistämällä merkille kalojen painon silmämääräisesti ennen orientaation muutosta. Testasin toimivuutta laskemalla kalojen lukumäärän ja painot orientaation muutoksien jälkeen. Kun lukemat pysyivät samoina kymmenen orientaation muutoksen ajan ilman ongelmia, katsoin tämän ominaisuuden toimivaksi.

4.5 Ruudunläpäisy

Testauksen kannalta ruudunläpäisyn testaaminen vaikutti työläimmältä, sillä silloin täytyi kuulua ruudunläpäisymusiikki sekä suorittaa seuraavan ruudun lataus. Sitten huomasin, että voin muuttaa peliä niin, että peli arpoo vain yhden pyydystettävän, ja niin ruudunläpäisyn testaus jatkui ilman suurta työmäärää (kuva 6). Testauksen läpäisykriteereiksi asetin, että ruudunläpäisyssä täytyi kuulua viimeisen kalan pyydystämisen tuulettelu ruudunläpäisymusiikin kanssa. Lopputulosten piti näkyä dialogissa, joka laittaisi pelin paussille, ja pelin pitäisi käynnistyä seuraavasta ruudusta dialogin sulkemisen jälkeen. Kymmenen perättäistä läpäisyä ilman ongelmia tulkittiin toimivaksi ominaisuudeksi.

KUVA 6. Kuvankaappaus lippoamispelistä.



5 YHTEENVETO

Työn tarkoitus oli luoda lippoamispeli Android-puhelimille. Suuri osa pelin vaatimuksista täyttyi. Liikeanturit ovat pelissä keskeisessä osassa. Puhelimen liikuttaminen vaikuttaa peliin reaaliajassa. Värinällä pelaaja saa tietoa siitä, mitä lipossa on pyydystettynä. Lipon etäisyyttä voi muuttaa. Pyydystettävän nosto voi onnistua tai epäonnistua. Verrattaessa vaatimuksiin lippoamispeliin jäi vielä puutteita. Pelissä ei ole 3D-grafiikkaa, eri pelimoodeja siian ja lohen lippoamiseen, osuutta jossa saavutaan kalastuspaikkaan, grafiikkaa Kukkolankoskelta, vaihtelevaa kosken pohjaa tai erilaisia virtauksia.

Peli toimii ja sitä on mielekäs pelata. Peli hyödyntää puhelimen liikesensoreita pelaamisessa ja värinää pyydystämistilanteissa. Pelissä on monipuolisemmin pyydystettävää, kuin aluksi vaadittiin. Pelissä on myös äänet tuomaan tunnelmaa ja saavutuksen tunnetta. Opin paljon uutta työn aikana. Liiketunnistus, äänien soittaminen ja pelisuunnittelu olivat muun muassa uusia asioita, joihin pääsin ensi kertaa tutustumaan lippoamispeliä tehdessäni.

Suurin syy puutteisiin on vaadittava työmäärä. Graafinen suunnittelu on mielenkiintoista, mutta itse opeteltuna harjoitteluun menee aikaa, ennen kuin saa aikaan tyydyttäviä tuloksia grafiikan suhteen.

Jatkokehitys on joiltain osin mahdollista, mutta ilman parempia grafiikoita ei esimerkiksi olisi järkevää tehdä kahta samankaltaista pelimoodia, joissa vain numerot ja tekstit ruudulla vaihtuisivat. Netin tai Bluetoothin avulla esimerkiksi kalahaaste, jossa piste-ennätys annetaan kaverin päihitettäväksi, voisi olla mielekäs tapa kisailla lippoamistaidoista älypuhelimilla.

LÄHTEET

1. Internetin käyttö mobiilia, laitteet henkilökohtaisia 2015. Tilastokeskus, Saatavissa: http://tilastokeskus.fi/til/sutivi/2015/sutivi_2015_2015-11-26_tie_001_fi.html. Hakupäivä 4.7.2016.

2. Smartphone OS Market Share, 2015 Q2 2015. International Data Corporation. Saatavissa: http://www.idc.com/prodserv/smartphone-os-market-share.jsp. Hakupäivä 28.6.2016.

3. Kuorikoski, Juho 2014. Sinivalkoinen pelikirja – Suomen pelialan kronikka 1984-2014. Saarijärvi: Fobos Kustannus