Saimaan ammattikorkeakoulu
Tekniikka Lappeenranta
Tietotekniikan koulutusohjelma
Organisaation IT-palvelut

Sami Hongisto

Mollan muistipeli

Opinnäytetyö 2014

Tiivistelmä

Sami Hongisto
Mollan muistipeli, 23 sivua, 1 liite
Saimaan ammattikorkeakoulu
Tekniikka Lappeenranta
Tietotekniikan koulutusohjelma
Organisaation IT-palvelut
Opinnäytetyö 2014
Ohjaajat: lehtori Yrjö Utti, Saimaan ammattikorkeakoulu, Timo Kainulainen, Saimaan mediakeskuksen päällikkö

Tässä opinnäytetyössä toteutettiin oppimispeli Googlen Android- sekä Applen iOS-käyttöjärjestelmille. Opinnäytetyöni asiakkaana toimi Saimaan mediakeskus.

Peli toteutettiin Java-ohjelmointikielellä käyttäen avoimen lähdekoodin libGDX-kirjastoa, joka kääntää Javalla kirjoitetun koodin eri alustoille yhteensopivaksi. Tämä mahdollisti helpon ja nopean toteutuksen ja testauksen.

Opinnäytetyön tuloksena saatiin valmis oppimispeli, joka julkaistiin Applen App Storessa sekä Googlen Google Playssa. Peliä voidaan käyttää opettamaan erilaisia muotoja, esineitä, ääniä sekä interaktiivisuutta tablettien ja tietokoneiden kanssa.

Asiasanat: Java, avoin lähdekoodi, mobiili, oppimispeli, Android, iOS

Abstract

Sami Hongisto
Molla's matching game, 23 pages, 1 appendix
Saimaa University of Applied Sciences
Technology Lappeenranta
Degree Programme in Information Techology
IT-services in Organisation
Bachelor´s Thesis 2014
Instructors: Senior Lecturer Yrjö Utti, Saimaa University of Applied Sciences,
Timo Kainulainen, Leader of Saimaan mediakeskus

The purpose of this thesis was to design and produce a learning game to Google's Android and Apple's iOS operating systems. The client for this project was Saimaan mediakeskus.

The game was produced by Java programming language with open source LibGDX framework which translates Java language so that iOS, Windows, Mac OS X and Android can understand it.

As a result of this thesis, the learning game got published in App Store and Google Play. It can be used to teach different kinds of shapes, objects, sounds and interactivity with tablets and computers.

Keywords: Java, open source, mobile, learning game, Android, iOS

Sisältö

Termit ja käsitteet

Apple	Yhdysvaltalainen yritys, joka suunnittelee, kehittää ja myy kulutuselektroniikkaa, ohjelmistoja ja tietokoneita
App Store	Applen kehittämä sovelluskauppa, jossa myydään sovelluksia Applen iOS-laitteisiin
Android	Googlen kehittämä käyttöjärjestelmä
Avoin lähdekoodi	Kehitysmenetelmä, jossa tekijä jakaa teoksensa muiden käytettäväksi ja muokattavaksi
COBOL	Common Business Oriented Language. Vuonna 1959 kehitetty ohjelmointikieli
Eclipse	Avoimen lähdekoodin ohjelmointiympäristö
Frame-by-frame	Animointityyli, jossa muutokset määritellään jokaiseen ruutuun erikseen.
Google	Google Inc. on yhdysvaltalainen yhtiö joka tarjoaa Internet palveluita, tuottaa ohjelmistoja sekä kehittää Android-käyttöjärjestelmää
Google Play	Googlen omistama digitaalinen sisältöpalvelu
Git	Suomalaisen Linus Torvaldsin kehittämä avoimen lähdekoodin versionhallintaohjelmisto
GitHub	Git-versionhallintaa käyttävä pilvipalvelu projekteille
iOS	Applen kehittämä käyttöjärjestelmä
Java	Ohjelmointikieli
JSON	JavaScript Object Notation. Yksinkertainen ja avoin tiedostomuoto tiedonvälitykseen
LibGDX	Java-pohjainen pelinkehitysalusta
Luokka	Olio-ohjelmoinnissa olion ominaisuuksien, menetelmien ja attribuuttien määritelmä, joka sisältää olion tiedot ja toiminnan
Metodi	Luokan tai olion aliohjelma, joka suorittaa tietyn tehtävän
Multi-platform	Järjestelmäriippumaton, toimii monella eri käyttöjärjestelmällä

PHP	PHP: Hypertext Preprocessor. Ohjelmointikieli joka soveltuu erityisesti web-sovelluskehitykseen
Spine 2D	2D-peleihin tarkoitettu animointityökalu
Texture Atlas	Suuri kuva, joka sisältää monta pienempää kuvaa
Texturepacker	Ohjelma, joka mahdollistaa kuvien pakkaamisen yhteen texture atlakseen
Tvt	Tieto- ja viestintätekniikka
UML	Unified Modeling Language. Graafinen mallinnuskieli järjestelmä- ja ohjelmistokehitystä varten

1 Johdanto

Tämä opinnäytetyö käsittelee libGDX:n avulla toteutettavan multi-platform-oppimispelin työvaiheita sekä työhön liittyviä tekniikoita ja työkaluja. Opinnäytetyö toteutetaan yhteistyössä Joonas Pirttiahon kanssa. Pirttiaho oli tutustunut Saimaan mediakeskuksen työntekijöihin aikaisemman projektin myötä, ja heille tuli puhetta, että jonkinlaiselle oppimispelille olisi tarvetta. Projekti osoittautui sopivaksi opinnäytetyön aiheeksi, joten päätös tehdä opinnäytetyö aiheesta syntyi talvella 2013. Oppimispelin rungoksi valikoitui muistipeli, jonka avulla lapset pystyvät opiskelemaan uusia asioita kuvien ja symbolien avustuksella. Oppimispeli on toteutettu osana Opetushallituksen ja Lappeenrannan kaupungin rahoittamaa ”eMolla materiaalia ja toimintaympäristö varhauskasvatukseen ja esiopetukseen” hanketta. Hanketta koordinoi Saimaan mediakeskus.

1.1 Tavoite

Opinnäytetyön tavoitteena on toteuttaa oppimispeli Android- sekä iOS-käyttöjärjestelmille käyttäen projektiin valittuja tekniikoita ja ohjelmistoja. Tavoitteemme oli oppia enemmän kyseisten tekniikoiden ja ohjelmistojen hyödyntämisestä sovelluskehityksessä.

1.2 Asiakas

”*Saimaan mediakeskus on Lappeenrannan kasvatus- ja opetustoimen alainen yksikkö. Se järjestää tieto- ja viestintätekniikan opetuskäytön täydennyskoulutusta opetushenkilöstölle ja tukee oppilaitoksia niiden tvt-hankkeissa. Saimaan mediakeskuksen tavoitteena on edesauttaa tieto- ja viestintätekniikan opetuskäytön ja mediakasvatuksen tueksi*”. (saimaanmediakeskus.fi, Mediakeskus)

Molla-hanke

Molla-hanke (Media, osallisuus, lapsi) käynnistyi vuonna 2011. Hanke saa rahoitusta Opetushallituksen oppimisympäristöjen kehittämisrahoituksesta. Tavoitteena on tuoda esiopetukseen sekä kaikenikäisille lapsille mediamaailma tutuksi turvallisessa ja osallistavassa ympäristössä.

eMolla-hanke

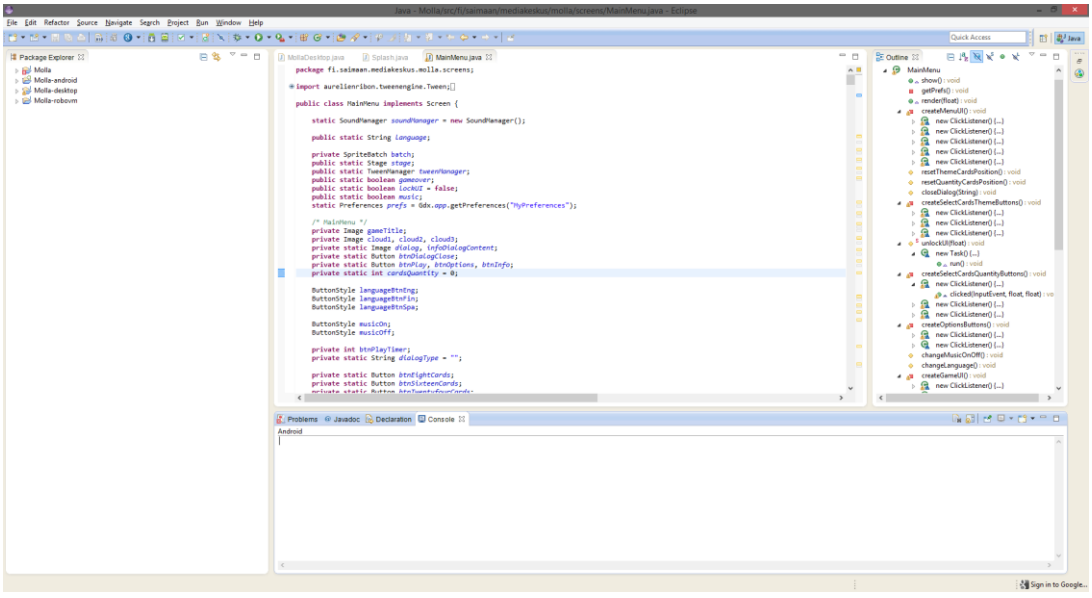
eMolla-hankkeessa toteutetaan oppimismateriaalia ja toimintamalleja, jotka soveltuvat lapselle, joka ei vielä osaa lukea. Ympäristö perustuu symboleihin, kuviin ja animoituihin ääniopasteisiin.

2 Työkalut ja tekniikat

Mobiilisovellusten kehityksessä on tarjolla monia erilaisia tekniikoita ja työkaluja. Tässä luvussa käydään läpi työkalut ja tekniikat, jotka valikoituivat projektiin soveltuviksi aikaisempien kokemusten sekä internetistä löytyneiden suosituksien avulla.

2.1 Eclipse

Eclipse (Kuva 1) on Java-pohjainen avoimen lähdekoodin ohjelmointiympäristö, joka on alunperin IBM:n kehittelemä. Eclipse julkaistiin avoimen lähdekoodin lisenssille vuonna 2003. Vuodesta 2004 lähtien Eclipsen kehityksestä on vastannut Eclipse Foundation-säätiö. Lisäosilla on mahdollista saada Eclipseen lisäominaisuuksia kuten UML-kaavioiden käsittelyä tai projektin versionhallintaa. Lisäosien ansiosta Eclipsellä voidaan ohjelmoida erittäin monella ohjelmointikielellä kuten C/C++, COBOL sekä PHP. (SearchSOA, Definition of Eclipse)



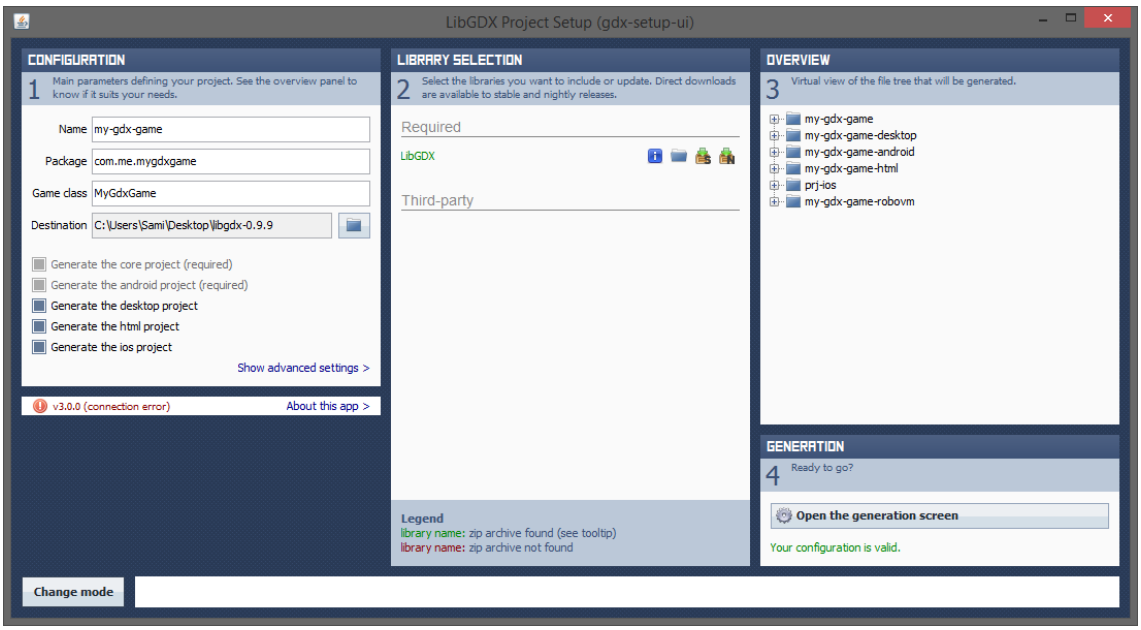
Kuva 1. Eclipsen käyttöliittymä

2.2 Java

Java-ohjelmointikieli on yleiskäyttöinen ja alustariippumaton korkean tason ohjelmointikieli, joka muistuttaa syntaksiltaan paljon C- ja C++-kieliä. Kyseisten kielten lisäksi se on saanut vaikutteita muistakin ohjelmointikielistä. Java on täysin oliopohjainen, koska kaikki ohjelmointi tapahtuu luokkien sisällä eikä kyseinen kieli salli globaaleita muuttujia tai itsenäisiä funktioita. Alustariippumattomuuden takia yhdellä alustalla toimiva koodi voidaan ajaa toisellakin alustalla ilman muutoksia. Java valikoitui projektin ohjelmointikieleksi LibGDX:n ja Eclipsen vaatimusten takia. (Tietokone.fi, Java ohjelmointikieli)

2.3 LibGDX

LibGDX on pelinkehitykseen tarkoitettu avoimen lähdekoodin alusta, joka on toteutettu Javalla, mutta suorituskykyyn liittyvät osat ovat kirjoitettu C- ja C++-kielellä paremman suorituskyvyn saavuttamiseksi (Kuva 2). LibGDX tarjoaa mahdollisuuden nopeaan kehittämiseen ja testaamiseen. Sovellus ei tarvitse kuin ohjelmoida kerran, jonka jälkeen se voidaan kääntää kaikille alustoille ilman alustariippuvaista koodia. (libGDX, Goals and Features)



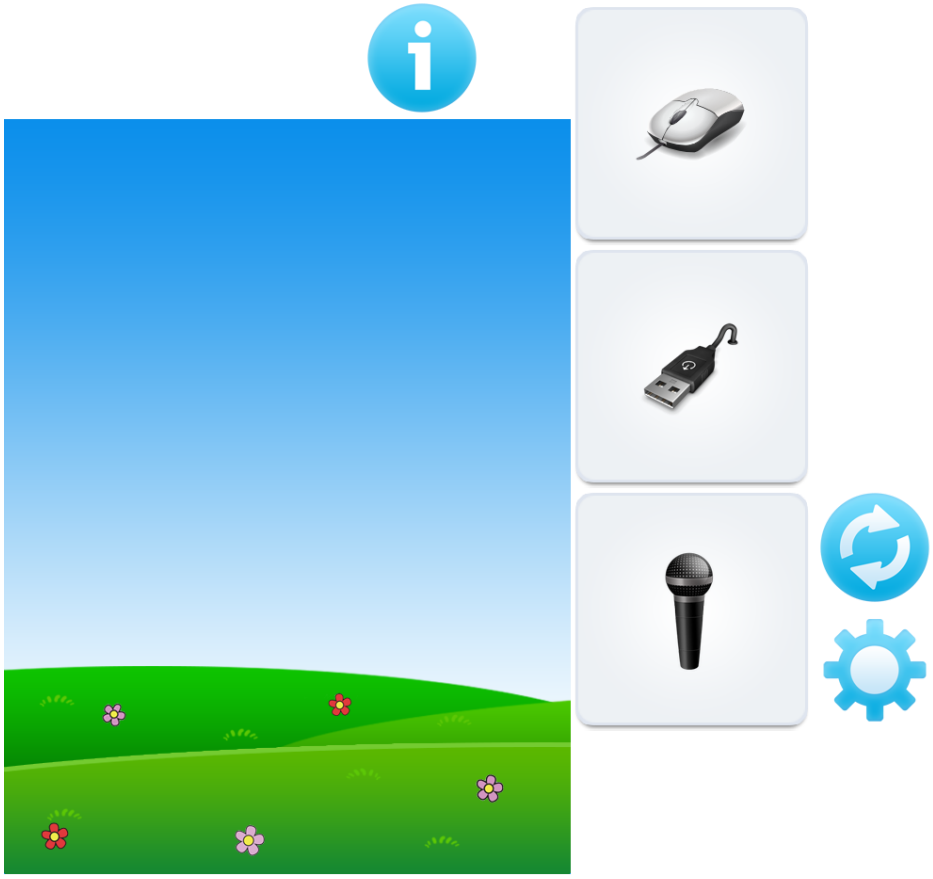
Kuva 2. LibGDX-projektin asetusruutu

2.4 Texturepacker

Texturepackerin avulla on mahdollista luoda monen pienen kuvan sijasta yksi iso kuva eli texture atlas. Tämä helpottaa huomattavasti kuvien käyttöä koodissa, samalla vapauttaen resursseja muuhun käyttöön ilman kuvien laadun karsimista. (CodeAndWeb, Features)

2.5 Texture Atlas

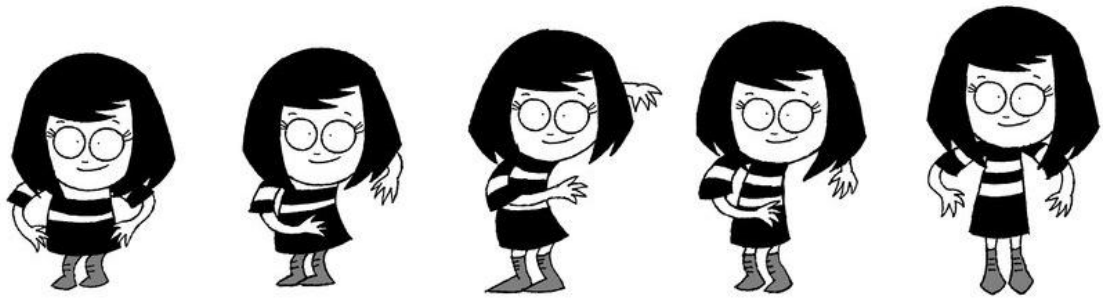
Tietokonegrafiikassa texture atlas (Kuva 3) tarkoittaa yhtä suurta kuvaa, joka sisältää pienempiä kuvia. Monta pienempää tekstuuria liitetään siis yhdeksi isoksi tekstuuriksi. Näitä pieniä tekstuureita kutsutaa alatekstuureiksi. Jokaisen alatekstuurin sijainti texture atlaksessa tallennetaan XY-koordinaateilla, jotta näitä pystytään käyttämään tarpeen vaatiessa. (Code.google.com, Texturepacker)



Kuva 3. Texturepackerin luoma texture atlas-tiedosto

2.6 Spine 2D

Spine 2D on 2D-peleihin tarkoitettu animointityökalu, joka tarjoaa monia hyötyjä verrattuna normaaliin frame-by-frame-animointiin. Spinen ansiosta animointiin tarvitaan vähemmän prosessointitehoa, sillä se tallentaa vain hahmon luiden tiedot eikä jokaiselle ruudulle omaa kuvaa. Normaalissa frame-by-frame-animoinnissahan jokaiselle ruudulle täytyy piirtää sama kuva pienillä muutoksilla, jolloin muodostuu liikettä, kun ruudut piirretään nopeasti peräjälkeen (Kuva 4). Spine tallentaa hahmosta yhden ison texture atlas-tiedoston, joka sisältää hahmon jokaisen luun (Kuva 5), sekä JSON-tiedoston, joka sisältää animaatioihin liittyvät luiden liikkeet. (Spine, Features)



Kuva 4. Esimerkki frame-by-frame animoinnista (Animation frames, DeviantArt)

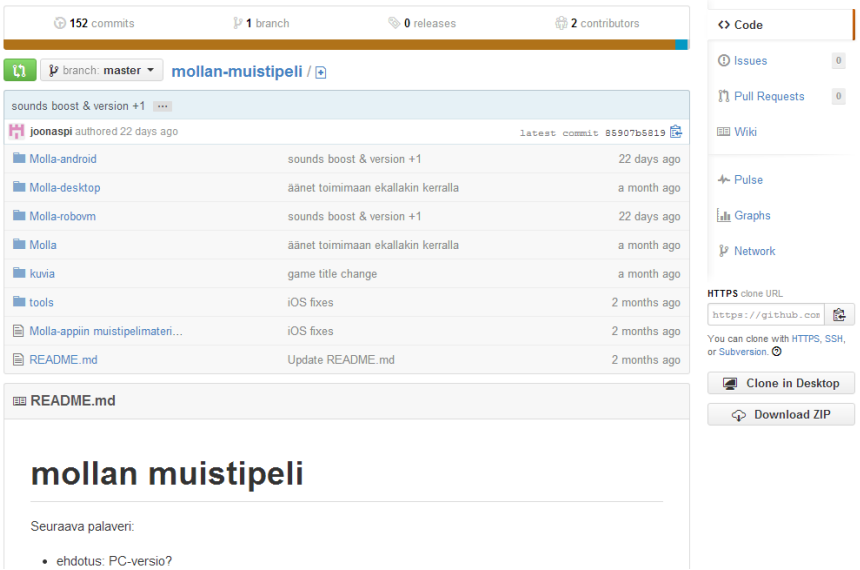
Kuva 5. Esimerkki Spinen animointikuvasta

2.7 GitHub

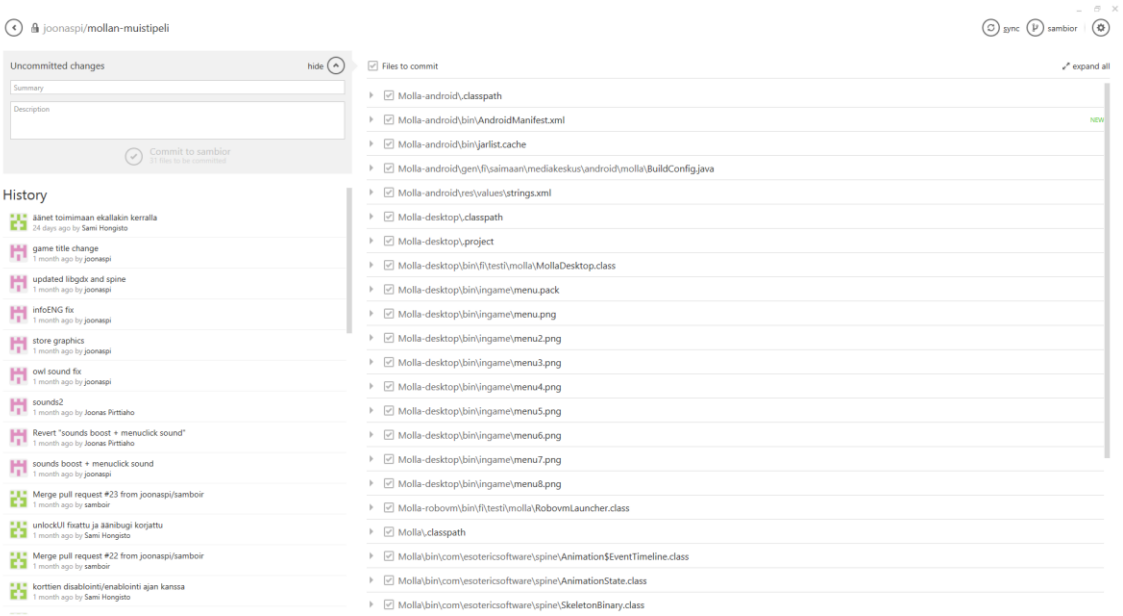
GitHub on internetpohjainen hostauspalvelu ohjelmistokehityksen projekteille, jotka käyttävät Git-versionhallintaa (Kuva 6). Vuonna 2011 GitHub oli suosituin koodisäilö avoimen lähdekoodin projekteille. GitHubin on mahdollista hallita kehitettävän ohjelmiston versioita ohjelmiston elinkaaren ajan. Samalla projekti on myös koko ajan varmuuskopioituna. GitHub mahdollistaa monen käyttäjän samanaikaisen koodauksen sekä koodin helpon yhtenäistämisen.

GitHub-palvelussa on mahdollista pitää avoimen lähdekoodin projekteja ilmaiseksi täysin avoimena. Kuka tahansa voi nähdä projektin lähdekoodit, lähettää vikailmoituksia tai päivityksiä lähdekoodiin. Maksua vastaan on mahdollista tehdä yksityisiä projekteja, joihin on rajattu pääsy vain halutuille käyttäjille. Opiskelijoiden on mahdollista saada tämä maksullinen jäsenyys ilmaiseksi kahden vuoden ajaksi.

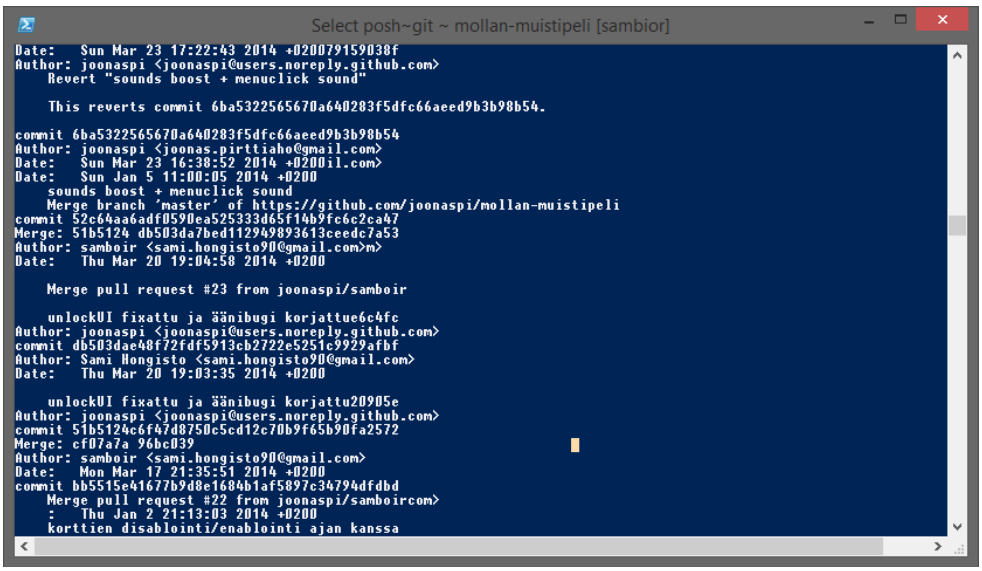
GitHub-ohjelmisto (Kuva 7) toimii Windows-, OS X-, sekä Linux-pohjaisilla käyttöjärjestelmällä. Git- ja GitHub-ohjelmistoja on mahdollista käyttää myös komentorivin kautta (Kuva 8), joten varsinaisia virallisia rautavaatimuksia ei ole ilmoitettu. Tähän voinee vaikuttaa myös se, että ohjelmistokehitykseen yleensä käytetään kehitysympäristöjä, joissa ohjelmistot ovat ajantasalla.



Kuva 6. GitHub-selainsovellus



Kuva 7. GitHub-työpöytäsovellus



Kuva 8. GitHub-komentokehote

3 Toteutus

Projektin aloitus tapahtui vuoden 2013 joulukuussa, jolloin aloimme Joonas Pirttiahon kanssa työstämään prototyyppiä mediakeskuksen ohjeiden mukaan. Prototyypin ideana oli näyttää, onko mahdollista tai kykenemmekö edes toteuttamaan ohjeiden mukaisen muistipelin Android- sekä iOS-käyttöjärjestelmille. Prototyypin vaatimuksena oli, että sen täytyi olla toimiva muistipeli ja sen täytyi toimia iOS- ja Android-laitteilla.

Valitsimme koodisäilöksi aikaseimpien kokemusten takia GitHub-ohjelmiston, jonka avulla pystyimme työstämään projektia samaan aikaan Pirttiahon kanssa ilman huolta koodin päällekkäin menemisestä. Samalla projektin varmuuskopiointi hoitui GitHubin avulla, sillä GitHub tallentaa automaattisesti aiemmat versiot ja niihin voi halutessaan palata.

LibGDX ja Eclipse valikoituivat projektimme työkaluiksi Pirttiahon aikaisempien kokemusten perusteella. Kyseiset ohjelmistot mahdollistivat meille nopean ja vaivattoman sovelluksen kehityksen ja testauksen. Pystyimme testaamaan sovellusta suoraan tietokoneella, koska libGDX tarjoaa projektin myös työpöytä ympäristöön, jolloin sitä voi testata Eclipsessä normaalin Java-ohjelman tavoin.

3.1 Prototyyppi

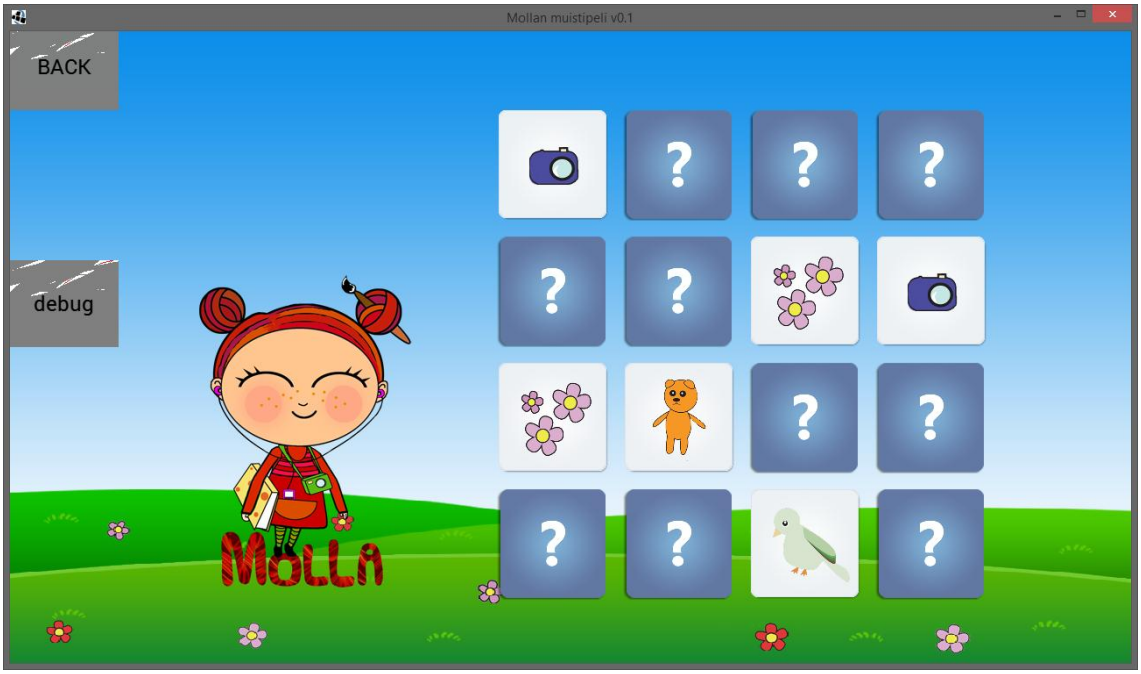
Päätimme toteuttaa prototyyppiin aloitusruudun, josta pääsi etenemään peliruutuun nappulaa painamalla. Peliruudussa (Kuva 10) pelaaja pystyi pelaamaan 16 kortin muistipelin, jonka suoritettuaan pelaaja palasi aloitusruutuun. Kun olimme saaneet pelin perustoiminnot toteutettua, pystyimme keskittymään pienempiin toimintoihin, kuten pelimaailman elävöittämiseen.

Halusimme tehdä muistipeliin joitain hienouksia, jotta peli ei olisi pelkkä staattinen ruutu, jossa kortit vain kääntyisivät, joten rupesimme miettimään, miten saisimme toteutettua peliin jotain elävyyttä. Koska pelin taustaksi oli valikoitunut kuva jossa on taivasta ja nurmikkoa, päädyimme toteuttamaan taustalle liikkuvia pilviä. Pilvien liikkumisen lisäksi päätimme, että Molla voisi myös liikkua ruudulla, mutta tuota ominaisuutta emme vielä prototyyppiin

toteuttanut. Tutkimustemme tuloksena päädyimme toteuttamaan Mollan animoinnin Spine 2D-ohjelmalla.

```
private void animateClouds() {

    // pilvien animointia, lasketaan random nopeus kun pilvi on poistunut ruudusta
    if (cloud1OnScreen) {
        cloud1X += cloud1Speed;
    }
    else {
        // MIN + math.random() * ((MAX - MIN) + 1)
        randomNumber = 5 + (int)(Math.random() * ((13 - 5) + 1));
        cloud1Speed = randomNumber;
        cloud1Speed = cloud1Speed / 10;
        cloud1OnScreen = true;
    }

    if (cloud2OnScreen) {
        cloud2X += cloud2Speed;
    }
    else {
        randomNumber = 5 + (int)(Math.random() * ((13 - 5) + 1));
        cloud2Speed = randomNumber;
        cloud2Speed = cloud2Speed / 10;
        cloud2OnScreen = true;
    }

    if (cloud3OnScreen) {
        cloud3X += cloud3Speed;
    }
    else {
        randomNumber = 8 + (int)(Math.random() * ((13 - 8) + 1));
        cloud3Speed = randomNumber;
        cloud3Speed = cloud3Speed / 10;
        cloud3OnScreen = true;
    }

    // liikutellaan pilvi+ruudulla
    cloud1.setPosition(stage.getWidth() - Gdx.graphics.getWidth()/5f + cloud1X, Gdx.graphics.getHeight()/1.5f);
    cloud2.setPosition(stage.getWidth() - Gdx.graphics.getWidth()/2.5f + cloud2X, Gdx.graphics.getHeight()/1.2f);
    cloud3.setPosition(stage.getWidth() - Gdx.graphics.getWidth()/1f + cloud3X, Gdx.graphics.getHeight()/1.3f);

    // tarkistetaan onko pilvi mennyt ruudusta yli
    if (cloud1.getX() > stage.getWidth()) {
        cloud1X = (int) stage.getWidth() * -1 - cloud1.getWidth();
        cloud1OnScreen = false;
    }

    if (cloud2.getX() > stage.getWidth()) {
        cloud2X = (int) stage.getWidth() * -1 - cloud2.getWidth();
        cloud2OnScreen = false;
    }

    if (cloud3.getX() > stage.getWidth()) {
        cloud3X = (int) stage.getWidth() * -1 - cloud3.getWidth();
        cloud3OnScreen = false;
    }
}
```

Kuva 9. Esimerkkikoodi jolla hoidetaan pilvien liikuttelua ruudulla



Kuva 10. Prototyypin peliruutu

3.2 Projektin aloitus

Saimaan mediakeskus hyväksyi prototyypin, jonka jälkeen rupesimme suunnittelemaan tarkemmin ohjelmiston ominaisuuksia. Palaverin aikaisen suunnittelun tuloksena sovimme, että Mollan täytyy puhua suomen lisäksi myös englantia sekä espanjaa. Espanja valikoitui kolmanneksi kieleksi, koska Lappeenrannan päiväkodeilla on ystävyyspäiväkoteja Espanjassa. Monikielisyyden lisäksi pelaajan täytyy pystyä valitsemaan erilaisia teemoja korteille sekä vaikeustaso. Korttien teemoiksi valikoituvat muodot, medialaitteet sekä ääni-eläin-kuvaparit. Mediakeskus hyväksyi ehdotuksemme Mollan animoinnista, joten rupesimme työstämään animointia sekä muita edellä mainittuja ominaisuuksia palaverin jälkeen.

Aloituspalaverin jälkeen toteutimme aloitusruudun ja peliruudun lisäksi korttien teemanvalintaruudun, josta pelaaja pystyy valitsemaan haluamansa teeman, sekä vaikeustasoruudun, josta pelaaja pystyy valitsemaan, millä määrällä kortteja hän haluaisi pelata. Vaihtoehtoina korttien määrälle on 8, 16 ja 24 tai 8 ja 16 riippuen siitä, millä korttien teemalla pelataan. Mollan puheäänet saimme mediakeskukselta, joten niiden suhteen ei tarvinut tehdä muuta kuin lisätä äänet sovellukseen. Äänet ladattiin Asset-managerin avulla pelin käyttöön (Kuva 11) ja tämän jälkeen ääniä pystyi käyttämään SoundManager-luokan playSound-metodilla. (Kuva 12)

```
private static void loadSounds() {
    soundGameCompleted = "sounds/success.mp3";
    manager.load(soundGameCompleted, Sound.class);

    soundMenuClick = "sounds/click.mp3";
    manager.load(soundMenuClick, Sound.class);

    musicBackground = "sounds/background_music.mp3";
    manager.load(musicBackground, Music.class);

    //molla suomi
    soundMollaAloitetaan = "sounds/molla/fin_aloitetaan.mp3";
    manager.load(soundMollaAloitetaan, Sound.class);

    soundMollaHeippaKaikille = "sounds/molla/fin_heippakaikille.mp3";
    manager.load(soundMollaHeippaKaikille, Sound.class);

    soundMollaHienoa = "sounds/molla/fin_hienoa.mp3";
    manager.load(soundMollaHienoa, Sound.class);

    soundMollaHienoHomma = "sounds/molla/fin_hienohomma.mp3";
    manager.load(soundMollaHienoHomma, Sound.class);

    soundMollaHyva = "sounds/molla/fin_hyva.mp3";
    manager.load(soundMollaHyva, Sound.class);

    soundMollaHyvaJuttu = "sounds/molla/fin_hyvajuttu.mp3";
    manager.load(soundMollaHyvaJuttu, Sound.class);

    soundMollaJee = "sounds/molla/fin_jee.mp3";
    manager.load(soundMollaJee, Sound.class);

    soundMollaJokoAloitetaan = "sounds/molla/fin_joko_aloitetaan.mp3";
    manager.load(soundMollaJokoAloitetaan, Sound.class);

    soundMollaOoh = "sounds/molla/fin_ooh.mp3";
    manager.load(soundMollaOoh, Sound.class);
```

Kuva 11. Äänien lataus ohjelman käyttöön

```
public static void playSound(Sounds sound){
    switch (sound) {
    case MENU_CLICK:
        Assets.manager.get(Assets.soundMenuClick, Sound.class).play(0.05f);
        break;
    case GAME_SUCCESS:
        Assets.manager.get(Assets.soundGameCompleted, Sound.class).play(0.05f);
        break;

    // molla suomi
    case MOLLA_ALOITETAAN:
        Assets.manager.get(Assets.soundMollaAloitetaan, Sound.class).play(1.0f);
        break;
    case MOLLA_HEIPPAKAIKILLE:
        Assets.manager.get(Assets.soundMollaHeippaKaikille, Sound.class).play(1.0f);
        break;
    case MOLLA_HIENOA:
        Assets.manager.get(Assets.soundMollaHienoa, Sound.class).play(1.0f);
        break;
    case MOLLA_HIENOHOMMA:
        Assets.manager.get(Assets.soundMollaHienoHomma, Sound.class).play(1.0f);
        break;
    case MOLLA_HYVA:
        Assets.manager.get(Assets.soundMollaHyva, Sound.class).play(1.0f);
        break;
    case MOLLA_HYVAJUTTU:
        Assets.manager.get(Assets.soundMollaHyvaJuttu, Sound.class).play(1.0f);
        break;
    case MOLLA_JEE:
        Assets.manager.get(Assets.soundMollaJee, Sound.class).play(1.0f);
        break;
    case MOLLA_JOKOALOITETAAN:
        Assets.manager.get(Assets.soundMollaJokoAloitetaan, Sound.class).play(1.0f);
        break;
    case MOLLA_OOH:
        Assets.manager.get(Assets.soundMollaOoh, Sound.class).play(1.0f);
        break;
```

Kuva 12. SoundManagerin toimintaa

3.3 Projektin lopetus

Kun olimme saaneet toteutettua kaikki suunnitellut ominaisuudet, oli aika pitää mediakeskuksen kanssa palaveri, jossa tarkastelimme, olivatko kaikki toiminnallisuudet kunnossa ja vastasiko muistipeli asiakkaan vaatimuksia. Todettuamme, että kaikki on kunnossa, peli toimii ja näyttää hyvältä, oli aika julkaista peli Google Playssa ja App Storessa.

Play-kauppa hyväksyi pelin muutaman tunnin sisällä, mutta App Storen kanssa kesti pidempään, noin viikon verran. Viikon odottelun jälkeen saimme Applelta viestin, jossa pyydettiin vaihtamaan Mollan muistipelin englanninkielinen nimi, Molla’s memory game, joksikin muuksi, koska saksalainen lautapelivalmistaja Ravensburger omistaa oikeudet sanan ”memory” käyttöön. Kyseisen episodin takia jouduimme vaihtamaan pelin englanninkielisen nimen Molla’s matching gameksi. Nimenmuutoksen jälkeen Apple hyväksyi myös Mollan muistipelin omaan sovelluskauppaansa.

4 Pelin esittely

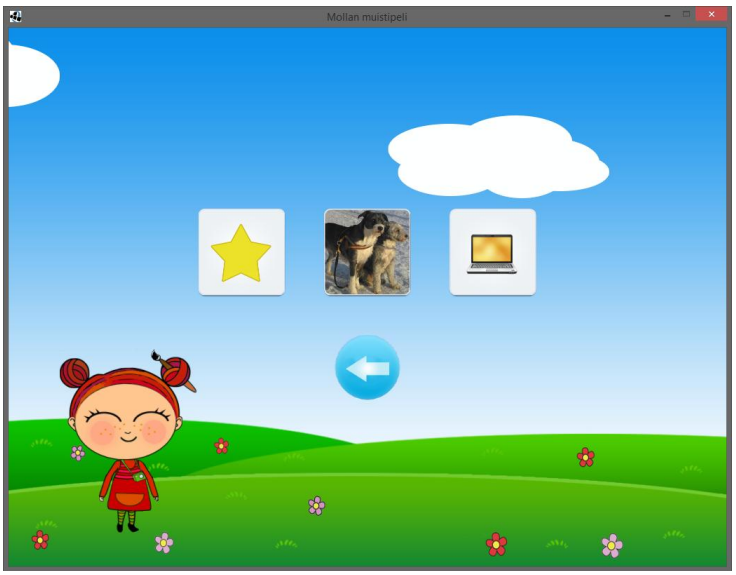
Kun pelaaja käynnistää sovelluksen, pääsee hän aloitusruutuun, jossa Molla tervehtii vilkuttamalla ja puhumalla (Kuva 13). Kyseisestä ruudusta löytyy pelaa-painikkeen lisäksi asetukset sekä inforuutu.

Kuva 13. Aloitusruutu, Molla tervehtii pelaajaa vilkuttamalla ja puhumalla

Asetuksista pystyy asettamaan taustamusiikin päälle tai pois päältä sekä valitsemaan pelin kielen joko englanniksi, suomeksi tai espanjaksi (Kuva 14).

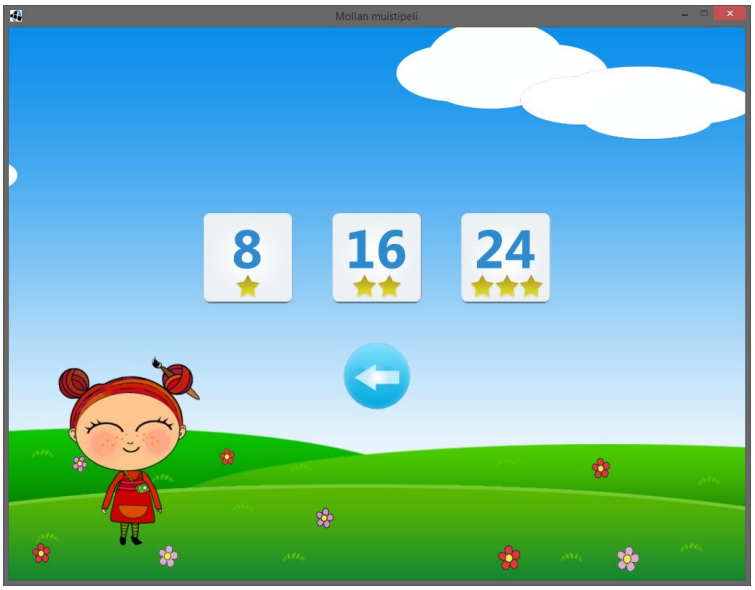
Kuva 14. Asetukset, kielen ja musiikin valinta

Pelaa-painiketta painamalla Molla pyytää pelaajaa valitsemaan haluamansa teeman muistikorteille. Valittavina teemoina on muodot, eläimet ääni-kuva-pareina sekä medialaitteet. Teemaruudusta pääsee takaisin aloitusruutun painamalla nuolinäppäintä (Kuva 15).



Kuva 15. Korttien teeman valintaruutu

Kun pelaaja on valinnut itselleen sopivan teeman, hän pääsee valitsemaan vaikeustason pelilleen. Peliin kuuluu kaksi tai kolme vaikeustasoa riippuen siitä, minkä teeman pelaaja valitsee. Vaikeustasoruudusta pelaaja pääsee takaisin teemanvalintaruutuun nuolta painamalla (Kuva 16).



Kuva 16. Korttien määrän valintaruutu

Vaikeustason valittuaan pelaaja etenee peliruutuun, jossa voi sitten pelailla muistipeliä Mollan kanssa. Kun pelaaja löytää parin tai kun hän on löytänyt kaikki parit, onnittelee Molla pelaajaa hienosta työstä (Kuva 17).

Kuva 17. Peliruutu, teemana medialaitteet

5 Yhteenveto

Opinnäytetyössä saatiin valmiiksi peli, joka toimii Android- sekä iOS-laitteilla. Pelistä julkaistiin myös samalla työpöytäversio, koska libGDX tarjosi kyseisen mahdollisuuden ilman muutoksien tekemistä koodiin. Peli toteutettiin käyttämällä ilmaisia mobiilikehitysohjelmistoja, jotka totesimme enemmän kuin riittäviksi projektin toteuttamiseen. LibGDX tarjoaa mahdottoman ison kokoelman erilaisia ominaisuuksia, joilla se helpottaa sovellusten tekemistä huomattavasti. Tämän ja helppokäyttöisyyden takia se onkin yksi suosituimmista sovelluskehityksen työkaluista. Kyseisen ohjelmiston avulla onnistuu varmasti myös paljon monimutkaisempienkin sovellusten teko ilman ongelmia.

Kuvat

Lähteet

Animation frames, DeviantArt
http://fc08.deviantart.net/fs70/i/2012/131/0/7/marilyn_animation_frames_by_nerdsman567-d4zepjs.jpg

CodeAndWeb, Features
http://www.codeandweb.com/texturepacker/features
Luettu 5.5.2014

Code.google.com, Texturepacker
https://code.google.com/p/libgdx/wiki/TexturePacker
Luettu 5.5.2014

LibGDX, Goals and Features
http://libgdx.badlogicgames.com/features.html
Luettu 5.5.2014

saimaanmediakeskus.fi, Mediakeskus
http://www.saimaanmediakeskus.fi/Mediakeskus
Luettu 5.5.2014

SearchSOA, Definition of Eclipse IDE
http://searchsoa.techtarget.com/definition/Eclipse
Luettu 5.5.2014

Spine, Features
http://esotericsoftware.com/spine-in-depth#Features
Luettu 5.5.2014

Tietokone.fi, Java ohjelmointikieli
http://www.tietokone.fi/artikkelit/java_ohjelmointikieli
Luettu 5.5.2014

```
//play button
// määritetään play nappulalle tyyli ja kuva
ButtonStyle btnPlayStyle = new ButtonStyle();
btnPlayStyle.up = new TextureRegionDrawable(Assets.btnPlayUp);
btnPlayStyle.down = new TextureRegionDrawable(Assets.btnPlayDown);
btnPlay = new Button(btnPlayStyle);

// ClickListener, eli mitä tapahtuu kun play-painiketta painetaan
btnPlay.addListener(new ClickListener(){
    @Override
    public void clicked(InputEvent event, float x, float y) {
        // tarkistetaan onko UI lukittuna vai ei
        if (!lockUI) {

            // jos UI ei lukittuna, lukitaan UI
            lockUI = true;

            // lukitaan aikaisemman ikkunan painikkeet
            disableMainMenuButtons();

            // enabloidaan seuraavan ikkunan painikkeet
            enableCardThemeWindowButtons();
            btnDialogClose.setTouchable(Touchable.enabled);

            // molla tekee animaation "hooray"
            molla.setAnimation("hooray");

            // molla sanoo käyttäjän valitsemalla kielellä, valitse peli
            Molla.SaySelectTheGame(language);

            // click-ääni
            soundManager.playSound(Sounds.MENU_CLICK);

            // palautetaan korttimäärä- sekä teemanvalinta-painikkeet alkuperäisille paikoilleen
            resetQuantityCardsPosition();
            resetThemeCardsPosition();
            btnMenuBackCardQuantity.setPosition(stage.getWidth() / 2 -btnMenuBackCardQuantity.getWidth() / 2,
            Gdx.graphics.getHeight() + Gdx.graphics.getHeight() / 5);

            btnMenuBackCardTheme.setPosition(stage.getWidth() / 2 -btnMenuBackCardTheme.getWidth() / 2,
            Gdx.graphics.getHeight() + Gdx.graphics.getHeight() / 5);

            // animoidaan alkuruudun tavarat pois ruudulta
            Tween.to(gameTitle, ActorAccessor.POSITION_XY, 1.5f).target(gameTitle.getX(), stage.getHeight()
            + gameTitle.getHeight()).ease(TweenEquations.easeInSine).start(tweenManager);

            Tween.to(btnPlay, ActorAccessor.POSITION_XY, 1.5f).target(btnPlay.getX(), 0)
            .ease(TweenEquations.easeInSine).start(tweenManager);

            Tween.to(btnPlay, ActorAccessor.ALPHA, 1).target(0).ease(TweenEquations.easeInSine)
            .delay(0.3f).start(tweenManager);

            Tween.to(btnOptions, ActorAccessor.POSITION_XY, 1).target(btnOptions.getX(),
            Gdx.graphics.getHeight()).ease(TweenEquations.easeInOutQuint).start(tweenManager);

            Tween.to(btnInfo, ActorAccessor.POSITION_XY, 1).target(btnInfo.getX(),
            -stage.getHeight() / 2f).ease(TweenEquations.easeInOutQuint).start(tweenManager);

            // animoidaan teemanvalintaruudun tavarat ruudulle
            Tween.to(btnSelectShapesTheme, ActorAccessor.POSITION_XY, 1.5f)
            .target(btnSelectAnimalsTheme.getX() - btnSelectShapesTheme.getWidth() * 1.4f
            , stage.getHeight()/2f)
            .ease(TweenEquations.easeInOutQuint).delay(0.0f).start(tweenManager);

            Tween.to(btnSelectAnimalsTheme, ActorAccessor.POSITION_XY, 1.5f)
            .target(Gdx.graphics.getWidth()/2f - btnSelectAnimalsTheme.getWidth() / 2f
            , stage.getHeight()/2f)
            .ease(TweenEquations.easeInOutQuint).delay(0.0f).start(tweenManager);

            Tween.to(btnSelectNormalTheme, ActorAccessor.POSITION_XY, 1.5f)
            .target(btnSelectAnimalsTheme.getX() + btnSelectNormalTheme.getWidth() * 1.4f
            , stage.getHeight()/2f)
            .ease(TweenEquations.easeInOutQuint).delay(0.0f).start(tweenManager);

            Tween.to(btnMenuBackCardTheme, ActorAccessor.POSITION_XY, 1.5f)
            .target(btnMenuBackCardTheme.getX(), Gdx.graphics.getHeight()/3.3f )
            .ease(TweenEquations.easeInSine).start(tweenManager);
```

```
            // enabloidaan UI takaisin käyttäjän käyttöön 1,5s kuluttua
            // estää samanaikaisen ja virheellisen nappuloiden painamisen
            unlockUI(1.5f);
        }
    }
});

// asetetaan pelaa-painikkeelle koko ja paikka sekä lisätään se stagelle
btnPlay.setSize(Gdx.graphics.getWidth()/9f, Gdx.graphics.getWidth()/10);
btnPlay.setPosition(stage.getWidth()/2.2f, Gdx.graphics.getHeight() + Gdx.graphics.getHeight() / 5);
stage.addActor(btnPlay);
```