

Jarno Lahti

Mobiilipelin tekninen toteutus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

15.9.2012

|  |  |  |
| --- | --- | --- |
| Tekijä(t)  Otsikko  Sivumäärä  Aika | | Jarno Lahti  Mobiilipelin tekninen toteutus  xx sivua + x liitettä  15.9.2012 |
| Tutkinto | | Insinööri (AMK) |
| Koulutusohjelma | | Tietotekniikan koulutusohjelma |
| Suuntautumisvaihtoehto | | Ohjelmistotekniikka |
| Ohjaaja(t) | | Tehtävänimike Etunimi Sukunimi  Tehtävänimike Etunimi Sukunimi |
|  | | |
| Avainsanat |  | |

|  |  |  |
| --- | --- | --- |
| Author(s)  Title  Number of Pages  Date | First name Jarno Lahti  Technical implementation of a mobilegame  xx pages + x appendices  15 September 2012 | |
| Degree | Bachelor of Engineering | |
| Degree Programme | Information Technology | |
| Specialisation option | Software Engineering | |
| Instructor(s) | First name Last name, Title (for example: Project Manager)  First name Last name, Title (for example: Principal Lecturer) | |
|  | | |
| Keywords | |  |

Sisällys

Lyhenteet

[1 Johdanto 1](#_Toc440625360)

[2 Mobiilipelien historiaa 1](#_Toc440625361)

[3 Java 2](#_Toc440625362)

[4 Insinöörityössä käytetyt työkalut 2](#_Toc440625363)

[4.1 Android studio 2](#_Toc440625364)

[4.2 LibGDX viitekehys 3](#_Toc440625365)

[4.3 Git-versionhallintajärjestelmä 7](#_Toc440625366)

[4.4 Asana –projektinhallintajärjestelmä 10](#_Toc440625367)

[5 SumTower 10](#_Toc440625368)

[5.1 Pelimuodot 10](#_Toc440625369)

[6 Pelin toteutus 10](#_Toc440625370)

[6.1 Projektin luonti 10](#_Toc440625371)

[6.2 Pelimekaniikka 10](#_Toc440625372)

[6.3 Käyttöliittymä 10](#_Toc440625373)

[7 Pohdinta 10](#_Toc440625374)

[Lähteet 11](#_Toc440625375)

Liitteet

Liite 1. Liitteen nimi

Liite 2. Liitteen nimi

Lyhenteet

APK Android application package. Tiedostomuoto jota Android käyttöjärjestelmä käyttää applikaatioiden jakamiseen ja asentamiseen

API Application Program Interface. Joukko työkaluja ja sääntöjä joita hyödynnetään sovelluskehityksessä

HTTP Hypertext Transfer Protocol. Protokolla tiedon lähetykseen ja vastaanottamiseen

HTTPS Hypertext Transfer Protocol Secure. Suojattu protkolla tiedon lähetykseen ja vastaanottoon

# Johdanto

Tähän kirjoitetaan johdanto. Käytä kirjoittamallesi tekstille tyyliä Leipäteksti. Käytä vastaavasti otsikoille tyylejä Otsikko 1, Otsikko 2 ja Otsikko 3. Kun käytät tyylejä, teksti muotoutuu automaattisesti oikein: riviväliksi tulee 1,5, molemmat reunat tasataan ja teksti tavutetaan. Kappaleen viimeisen rivin ei tarvitse ulottua oikeaan reunaan asti.

Uusi kappale alkaa vasemmasta reunasta.

# Mobiilipelien historiaa

Mobiilipelien historia alkoi vuodesta 1997, jolloin suomalainen matkapuhelinyhtiö Nokia julkaisi Nokia 6110 –matkapuhelimen. Tähän matkapuhelimeen oli esiasennettu Snake, joka tunnetaan myös nimellä matopeli. Matkapuhelimissa näytöt olivat silloin hyvin pieniä ja kaksivärisiä, joten peli ei ollut mitenkään graafisesti hieno, vaan yksinkertaisia mustista pikseleistä koostuvia muotoja vihreällä taustalla. Pelin tavoite oli niinkin yksinkertainen, että pelaajan täytyi ohjata pelihahmona toimivaa matoa saaliiden luo. Peli loppui jos mato törmäsi seinään tai itseensä. Mato kasvoi jokaisen syödyn saaliin jälkeen, joka toi peliin lisää vaikeusastetta kokoajan, koska madon kasvaessa pelialue käytännössä pieneni. [1.]

Kuva 1 Pelikuvaa matopelistä



2000-luvun alkupuolella kuluttajamarkkinoille alkoi saapua värinäytöllisiä matkapuhelimia, joten mobiilipelaaminen koki uuden merkittävän käänteen, kun pelit saivat syvyyttä värien kautta. Toinen merkittävä asia, joka kiihdytti mobiilipelien kehittymistä Java 2 Micro Edition (J2ME) ja se että uudet laitteet alkoivat tukea sitä[1]. J2ME on sulautetuille järjestelmille tarkoitettu versio Javasta. J2ME mahdollisti sen, että samaa koodia pystyi ajamaan useilla eri laitteilla ja jopa eri valmistajien laitteilla. Pelejä ja ohjelmia pystyi myös kehittämään ja testaamaan työpöytäympäristössä, joka nopeutti kehitystyötä. [2.]

Kuva 2 Pelikuvaa Jamdat Bowling pelistä

# Java

# Insinöörityössä käytetyt työkalut

## Android studio

Android Studio on virallinen ohjelmointiympäristö Android -applikaatioille ja se perustuu IntelliJ IDEA Java -ohjelmointiympäristöön. Se tarjoaa kehittäjälle useita työkaluja, kuten esimerkiksi:

* Gradleen perustuvan koontijärjestelmän
* Koontivariaatioita ja moni-APK tiedosto generaation
* Valmiita koodipohjia, jotka auttavat yleisten toimintojen tekemisessä
* Sommittelueditorin, jossa on tuettuna raahaa ja pudota –teemaeditointi
* Lint –työkaluja joiden avulla voidaan napata suorituskykyyn, käytettävyyteen, yhteensopivuuteen ja muihin ongelmiin liittyviä virheitä
* ProGuard ja applikaation signeeraus
* Sisäänrakennettu tuki googlen pilvipalveluille.

Android studio tarjoaa myös monia muita hyödyllisiä työkaluja. [3.]

**//TODO: lisää asiaa android studiosta**

## LibGDX viitekehys

Libdgx on avoimeen lähdekoodiin perustuva järjestelmäriippumaton peli- ja applikaatioviitekehys, joka tukee Windows, Linux, Mac OS X, Android, Blackberry, iOS ja HTML5 alustoja. Libgdx mahdollistaa sen, että kehittäjä voi kirjoittaa koodia ja se kääntyy kaikille yllä mainituille alustoille ilman minkäänlaisia muutoksia koodiin. Tämä mahdollistaa nopean pelin kehityksen, koska kehittäjä voi käytännössä ohjelmoida ja testata peliä, joka on tarkoitettu pelattavaksi esimerkiksi mobiililaitteella, täysin työpöytäympäristössä. Libgdx käyttää Javaa ohjelmointikielenä ja se antaa kehittäjälle käyttöön koko Javan tarjoaman ekosysteemin, mikä mahdollistaa kehittäjän olla luova. Libgdx tähtää siihen, että se ei olisi pelimoottori vaan lähempänä viitekehystä. Se antaa kehittäjälle vahvat työkalut, joista valita ja antaa kehittäjän itse päättää, miten haluaa pelin tai applikaationsa kirjoittaa. [4.]

Liittessä 1 on esitettynä Android- ja työpöytäsovelluksen käynnistysluokat. Nämä luokat sisältävät mahdolliset alustariippuvaiset alustukset ja asetukset. Esimerkiksi luokassa DesktopLauncher alustetaan ensimmäisenä olio config, joka on tyyppiä LwjglApplicationConfiguration. Tälle oliolle voitaisiin sitten määrittää esimerkiksi applikaation ruudun leveys ja korkeus pikseleissä tai vaikka asettaa applikaatio kokonäyttötilaan. Kun mahdolliset asetukset on määritetty, siirrytään itse applikaation applikaation avaamiseen. Työpöytäsovelluksella luodaan uusi LwjglApplication -tyyppiä oleva instanssi, jolle annetaan parametreiksi edellä mainitty config -olio ja uusi instanssi projektin pääluokasta, jossa sijaitsee applikaation toimintalogiikka. Tämä olio sitten käynnistää applikaation käyttäjälle näkyvän osuuden eli tässä tapauksessa peliruudun, joka on esimerkiksi leveydeltään ja korkeudeltaan edellä mainitun config –olion määritysten mukainen.

Libgdx on avoin viitekehys, sillä se antaa kehittäjän käyttää alhaisen tason toimintoja, kuten tiedostojärjestelmiä, syöttölaitteita, audiolaitteita sekä OpenGL:ää yhdistetyn OpenGL ES 2.0 ja 3.0 rajapinnan kautta. Näiden alhaisen tason laitteiden päälle on rakennettu useita sovellusrajapintoja, joiden avulla yleiset toiminnot, kuten esimerkiksi spritejen ja tekstin renderöinti, käyttöliittymien rakentaminen, musiikin toistaminen, erilaisten matemaattisten funktioiden suorittaminen ja eri tietotyyppien jäsentely, onnistuu vaivatta. [4.]

Libgdx tekee tarvittaessa kaikki natiiviin koodiin liittyvät toiminnot itsenäisesti ja ne suoritetaan yleensä koodin käännösvaiheessa, jolloin kehittäjän ei tästä tarvitse välittää [3]. Natiivilla koodilla tarkoitetaan koodia, joka on suunniteltu ajettavaksi määrätynlaisella prosessorityypillä. Natiivia koodia ei siis pysty ajamaan muilla prosessoreilla ellei koodia ole tarkoitettu emuloitavaksi. [5.]

Libgdx –sovelluksen elinkaari koostuu viidestä eri vaiheesta, jotka ovat luonti(create), pysäytys(pause), jatkaminen(resume), renderöinti(render) ja hävittäminen(dispose). Näihin ohjelman vaiheisiin kehittäjä pääsee käsiksi implementoimalla applikaationsa pääluokkaan libgdx:n oman rajapinnan *ApplicationListener*. ApplicationListener rajapinta sisältää metodit create, resize, render, pause, resume ja dispose, joiden avulla kehittäjä pääsee käsiksi sovelluksen elinkaareen. [6.] Metodien käyttötarkoitukset ovat seuraavat:

* **Create:** Tätä metodia kutsutaan vain kerran sovelluksen käynnistyksen yhteydessä, joten siinä voi tehdä esimerkiksi sovellukseen liittyvät initialisaatiot
* **Resize:** Tätä metodia kutsutaan heti createn jälkeen ja myös aina kun ohjelman ikkunana kokoa muutetaan esimerkiksi työpöytäsovelluksessa. Tämä metodi ottaa vastaan kaksi int -tyyppistä parametriä, jotka ovat pikselileveys ja –korkeus.
* **Render:** Tätä metodia kutsutaan sovelluksen pääsilmukassa aina kun piirtämistä kuuluu tapahtua. Tätä metodia käytetään yleensä myös esimerkiksi pelilogiikan päivittämiseen.
* **Pause:** Kaikilla sovellusalustoilla tätä metodia kutsutaan, kun sovellusta ollaan lopettamassa. Jos sovellusalustana on Android, niin tätä metodia kutsutaan, kun sovellus halutaan pysäyttää. Tälläisiä tilanteita ovat esimerkiksi, kun käyttäjälle tulee puhelu tai käyttäjä painaa laitteen koti –näppäintä.
* **Resume:** Tätä metodia kutsutaan pelkästään Androidilla. Kutsu tapahtuu pause –metodin jälkeen, kun sovellukseen ollaan palaamassa esimerkiksi, kun käyttäjä lopettaa puhelun, joka keskeytti sovelluksen aikaisemmin.
* **Dispose**: Tätä metodia kutsutaan kun sovellus ollaan lopettamassa. Ennen dispose -metodin kutsumista kutsutaan pause –metodia.



Kuva 3 Libgdx –sovelluksen elinkaari suomeksi havainnollistettuna [6].

Libgdx:n ydin koostuu kuudesta eri moduulista, jotka tarjoavat keinon kommunikoida käyttöjärjestelmän kanssa. Näitä moduuleja voidaan kutsua kaikissa järjestelmissä ja ne toimivat kaikissa samalla tavalla. [6.] Moduulit ovat seuraavat:

* **Sovellus(Application):** Suorittaa applikaation ja pitää huolen siitä, että API on tietoinen applikaation tason tapahtumista, kuten esimerkiksi ikkunan koon muutoksista.
* **Tiedostot(Files):** Mahdollistaa tiedostojärjestelmän käytön eri alustoilla.
* **Syöttö(Input):** Informoi API:a käyttäjän syöttämistä komennoista, kuten hiiren klikkauksista ja näppäimistön tai kosketusnäytön painalluksista.
* **Verkko(Net):** Tarjoaa tavat päästä käsiksi resursseihin HTTP tai HTTPS protokollia hyödyntäen.
* **Ääni(Audio):** Tarjoaa tavat toisaa ääniefektejä ja musiikkia. Mahdollistaa myös pääsemisen äänilaiteisiin käsiksi koodissa.
* **Grafiikka(Graphics):** Mahdollistaa käsiksi pääsemisen OpenGL ES 2.0 –rajapintaan.

Näihin moduuleihin pääsee käsiksi koodissa Gdx –luokan kautta, jossa ne ovat staattisina muuttujina. [7.] Seuraavassa kuvassa havainnollistan kuinka näihin moduuleihin pääsee käsiksi koodissa.

Esimerkkikoodi 1 Moduulien alustus

**//TODO: lisää asiaa libGDX:stä**

## Git-versionhallintajärjestelmä

Versionhallinta on tärkeä osa-alue ohjelmistokehityksessä, koska sillä helpotetaan tiimityöskentelyn toimintaa huomattavasti. Versionhallinnalla tarkoitetaan yleensä työkalua tai menetelmää, jolla pidetään projektissa olevat tiedostot ajan tasalla ja virhetilanteiden sattuessa se mahdollistaa myös paluun edelliseen versioon. Pienissä ohjelmistoprojekteissa versionhallinta voidaan toteuttaa jopa niin yksinkertaisesti, että tiedostoja kopioidaan paikallisesti vain uusiin kansioihin talteen ja esimerkiksi nimetään kansio versionumerolla. Tämä toimii tiettyyn pisteeseen saakka, mutta on erittäin herkkä ongelmatilanteille, kuten esimerkiksi kovalevyn rikkoutumiselle tai muuten vaan datan katoamiselle. [8, s.1]

Git on versionhallintajärjestelmä, joka perustuu hajautettuun versionhallintamalliin. Hajautetulla versionhallinnalla tarkoitetaan sitä, että projektin tiedostoilla on jokin keskitetty tietokanta, josta jokainen käyttäjä kopioi koko tietokannan paikallisesti omalle työkoneelleen. Käyttäjät tekevät muutoksia näihin paikallisiin tiedostoihin ja lopulta lähettävät itse tekemät muutokset keskitettyyn tietokantaan. Koska koko tietokannan sisältö on jokaisen omalla työkoneella ja kaikki muutokset tehdään paikallistiedostoihin, mahdollistaa tämä helpon tavan palauttaa varmuuskopio tietokannasta, jos esimerkiksi palvelin hajoaa. [8, s.4]



Kuva 4. Hajautetun versionhallinnan havainnollistamiskaavio.

Git:in toiminta perustuu kolmeen eri vaiheeseen, joissa tiedostot voivat olla. kommitoituna(commited), modifioituina(modified) tai esitetty(staged). Kommitoitu tarkoittaa, että data on tallennettu paikalliseen tietokantaan. Modifioitu tarkoittaa, että tiedostoissa on tapahtunut muutoksia, mutta niitä ei ole vielä kommitoitu. Esitetty tarkoittaa sitä että muokatuista tiedostoista on otettu talteen nykyinen tila ja määritetty se menemään seuraavassa tallennuksessa paikalliseen tietokantaan. Tämä tarkoittaa sitä, että git -projektissa on kolme eri lohkoa, joissa tiedostot ovat: Git -kansio, työkansio ja esitysalue. Git-kansio sisältää metadatan ja projektin tietokannan. työkansio sisältää kopiot tietokannan tiedostoista, joita käyttäjä muokkaa. Esitysalue on git-kansion sisällä oleva tiedosto, joka sisältää informaatiota seuraavaan kommittiin menevästä datasta. [8, s.8]



Kuva 5. Gitin kolme päätilaa ja niissä tiedostojen liikkuminen.

Työskentely giti:llä tapahtuu yleensä komentorivin kautta, mutta vaihtoehtoisia graafisella käyttöliittymällä varustettuja ohjelmistoja on myös saatavilla. Näissä graafisissa ohjelmistoissa on yleensä ongelmana se, että niillä ei voi tehdä kaikkea mahdollista, mitä komentorivillä työskentelyllä voi tehdä. Tämä johtuu siitä, että graafiset ohjelmistot usein tarjoavat käyttäjälle vain osan suoritettavista komennoista. [8, s.9]

//kuva gitin komentorivistä

## Asana –projektinhallintajärjestelmä

# SumTower

## Pelimuodot

# Pelin toteutus

## Projektin luonti

## Pelimekaniikka

## Käyttöliittymä

# Pohdinta

Lähteet

1. Phone Arena. 2011. History of mobile gaming. Verkkodokumentti. <http://www.phonearena.com/news/History-of-mobile-gaming\_id17949> Luettu 15.1.2016
2. Java World. 2003. Develop state-of-the-art mobile games. Verkkodokumentti. < http://www.javaworld.com/article/2073796/mobile-java/mobile-java-develop-state-of-the-art-mobile-games.html> 7.11.2003. Luettu 3.2.2015.
3. Android developers. 2015. Android Studio Overview. Verkkodokumentti. <http://developer.android.com/tools/studio/index.html> Luettu 2.1.2016.
4. Github. 2015. Libgdx. Introduction. Verkkodokumentti. <https://github.com/libgdx/libgdx/wiki/Introduction> 29.7.2015. Luettu 2.1.2016.
5. Techopedia. 2015. Native code. Verkkodokumentti. <https://www.techopedia.com/definition/3846/native-code> Luettu 2.1.2016.
6. Github. 2015. Libgdx. The life cycle. Verkkodokumentti. <https://github.com/libgdx/libgdx/wiki/The-life-cycle> Luettu 14.1.2016
7. Github. 2015. Libgdx. The application framework. Verkkodokumentti. <https://github.com/libgdx/libgdx/wiki/The-application-framework> Luettu 4.1.2016
8. Chacon, Scott & Straub, Ben. 2014. Pro Git. Apress.

**Android- ja työpöytäsovelluksen sekä pääohjelman käynnistysluokat**

**Liitteen otsikko**