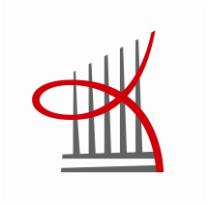


# MIND READER -PROJEKTIN OHJELMISTOARKKITEHTUURI

Lauri Nykänen

Opinnäytetyö
Joulukuu 2013
Tietotekniikka
Ohjelmistotekniikka

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

**TIIVISTELMÄ**

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto

NYKÄNEN, LAURI:
Mind Reader -projektin ohjelmistoarkkitehtuuri

Opinnäytetyö 26 sivua, joista liitteitä 3 sivua
Joulukuu 2013

---

Opinnäytetyössäni käsittelen Demolassa syksyllä 2012 tehdyn Mind Reader –projektin ohjelmistoarkkitehtuuria, projektin toteutusta ja projektiin liittyvää liikkeentunnistustekniikkaa.

Liikkeentunnistustekniikka –luvussa kerron liiketunnistustekniikan taustasta ja liikkeentunnistuskameroista keskityn vain niihin, jotka liittyivät Mind Reader -projektiin läheisesti. Nämä kamerat ovat Asus Xtion –kamera ja Microsoft Kinect –kamera.

Projektin suunnittelusta kerron ensinnäkin projektin lähtökohdista ja sen tavoitteista. Tavoitteena oli kahden ohjelman tekeminen. Ensimmäinen ohjelma oli datansyöttöohjelma, jolla pystyttäisiin opettamaan ohjelmalle erilaisten käyttäjän ilmeiden tunnistamista. Toisen ohjelman oli tarkoitus hyödyntää datansyöttöohjelman keräämää dataa. Tästä ohjelmasta oli tarkoitus tehdä peliohjelma, jossa käyttäjää kehotettaisiin tekemään erilaisia ilmeitä ja liikkeitä.

Ohjelman ohjelmistoarkkitehtuurista kerron sen neljästä komponentista. Liikkeentunnistamiseen käytetään Asus Xtion –kameraa. Java-ohjelma suorittaa liikkeentunnistusta OpenNI-kehyksen avulla. Web-kamera tallentaa kuvia jatkuvasti. Näistä kuvista oli tarkoitus tunnistaa kasvojen ilme, mutta ominaisuus ei päätynyt lopuliseen ohjelman versioon. Java-ohjelma generoi käyttäjän kehon pisteistä SVG-kuvaan, josta generoidaan PNG-kuva. Java-ohjelman tallentamat tiedot lähetetään käyttöliittymälle Glassfish-sovelluspalvelimen avulla.

Projektin lopputuloksena projektista valmistui liikkeentunnistuspeli, jossa käyttäjää kehoitetaan suorittamaan kuutta erilaista liikettä. Ohjelmasta jouduttiin karsimaan ominaisuuksia, sillä projektipäällikön puuttuminen johti erinäisiin ongelmiin.

---

Asiasanat: liikkeentunnistus, ohjelmistoarkkitehtuuri, konenäkö

**ABSTRACT**

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Option in Software Engineering

NYKÄNEN, LAURI:
Software architecture of Mind Reader project

Bachelor's thesis 26 pages, appendices 3 pages
December 2013

---

In my thesis I will talk about software architecture of a project made in Demola on autumn of 2012. I will also talk about project's design process and about motion sensing technology.

In the motion sensing chapter I tell about background of motion detection technology. I only focus on those motion sensing cameras that were closely related to the Mind Reader project. These cameras were Asus Xtion and Microsoft Kinect.

In the next chapter I will go through the project design process, starting from project's background and project's goal. The project aimed to create two programs. First program would be a data collection program, which would be used to train the program to recognize emotions from user's face. A second program would make a use of other program's recognition capabilities. This program would be a game, which would encourage the player to make different face expressions and movements.

In the software architecture chapter I will focus on the four main components of the product. Asus Xtion camera is used for motion detection. Java program performs the motion detection with OpenNI framework. A webcam saves images all the time. These images were meant to be used by the face recognition section but the feature never made it to the final product. Java program generates an SVG image from user's body part points and saves it as a PNG image. The files which Java program saves are sent to user interface with a Glassfish application server.

The end result of this project was a motion sensing game in which user is encouraged to perform six different movements. Many features had to be cut because of many difficulties caused by project not having a project leader.

---

Keywords: motion sensing technology, software architecture, computer vision

**SISÄLLYS**

## LYHENTEET JA TERMIT

| | |
|---|---|
| HTML | Verkkosivujen kuvaamiseen käytettävä merkintäkieli |
| JavaCV | OpenCV:n rajapinta Javalle |
| JavaScript | Verkkosivuissa paljon käytetty skriptikieli |
| OpenCV | Kirjasto, joka tarjoaa funktiot konenäköä varten |
| OpenNI-kehys | Ohjelmointirajapintojen kokoelma liikkeentunnistukseen |
| PNG | Portable Network Graphics, kuvan tallennusformaatti |
| SVG | Scalable Vector Graphics, vektorikuvien kuvauskieli |
| VGA | Video Graphics Array, näyttöstandardi |

# 1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on tutustua Demolassa tehdyn Mind Reader -projektin ohjelmistoarkkitehtuuriin. Projektin asiakkaana toimi Tampereen ammattikorkeakoulu ja se totetutettiin syksyllä 2012.

Koneet eivät ymmärrä ihmisten elkeitä. Projektimme tilattiin tätä puutetta paikkaamaan. Ohjelman toteutuksen reunaehtoina oli vain se, että saamme toteutettua aiheesta demon reilun kolmen kuukauden määräaikaan mennessä.

Projektin tarkoituksena oli tunnistaa ihmisen kasvoista henkilön tunne ja tämän rinnalle toteuttaa tunnistusta hyväksikäyttävä peli. Projekti ei saavuttanut kuitenkaan täysin tätä tavoitetta, vaan lopputuloksena saatiin webbipohjainen sovellus, jossa tunteiden sijaan tunnistetaan ihmisen kehon liikkeitä.

Projektia toteuttava ryhmämme koostui kolmesta henkilöstä: graafisesta suunnittelijasta, käyttöliittymäsuunnittelijasta ja ohjelmistosuunnittelijasta, joka oli minun roolini ryhmässä. Ryhmässämme oli neljäskin jäsen, joka kuitenkin jätti projektin alkuvaiheessa. Vastuutehtäväni kasvoivat ryhmän pienenemisen vuoksi. Lopulta toteutin java-pohjaisen liiketunnistuspelin ja sitä hyödyksi käyttävän internet-selaimessa ajettavan pelin.

Tässä opinnäytetyössä tarkastellaan projektin ohjelmistoarkkitehtuuria. Yritän hahmottaa miksi tässä projektissa päädyttiin tietynlaisiin ratkaisuihin ongelmien ratkaisemiseksi.

# 2 LIIKETUNNISTUSTEKNIIKKA

Liiketunnistus oli tärkeä osa tätä projektia. Kamerana käytettiin Asus Xtionia ja tunnistus suoritettiin käyttämällä hyväksi PrimeSensen NITE-moduulia (PrimeSense NITE, 2013). NITE-moduuli on yksi osa OpenNI-kehystä ja se sisällyttää algoritmit liikkeentunnistukseen. Tässä luvussa kerrotaan lyhyesti liiketunnistustekniikan historiasta ja projektissa käytetystä ja siihen läheisesti liittyvästä liiketunnistuskamerasta.

## 2.1 Liiketunnistustekniikan tausta

Liikkeentunnistustekniikka on tullut konsoleihin kivisen tien kautta. Aivan ensimmäiset yritykset liikkeentunnistustekniikan tuomisesta koteihin olivat 90-luvulla Segan laitteilla SegaPods ja Sega Activator. 2000-luvulla kuudennen konsolisukupolven konsoleilla alkoi jo olla vähän enemmän yritystä Sonyn PlayStation 2 –laitteelle tehdyllä PlayStation EyeToy -laitteella. Seitsemännen konsolisukupolven laitteilla oli jo reilusti kehittyneempää tekniikkaa Microsoft Kinect- ja PlayStation Eye –laitteilla.

## 2.2 Liiketunnistuslaitteet

Liiketunnistukseen kykenevät laitteet ovat tulleet markkinoille viime vuosina. Laitteet ovat yleistyneet olohuoneisiin pelikonsolien mukana. Liiketunnistimet ovat yleistyneet ihmisten olohuoneisiin osaksi viihdelaitteiden toimintaa.

### 2.2.1 Microsoft Kinect -laite

Microsoft julkaisi vuonna 2010 Xbox 360 -konsolillensa Kinect-nimisen liikkeentunnistuslaitteen. Kinectin kamera näkee ympäristönsä VGA-videostandardin avulla, eli resoluutiona on 640x480 pikseliä (Melgar, E. Díez, C. 2012). Microsoft julkaisi vuonna 2011 kehittäjille suunnatun Windows-yhteensopivan version Kinectistä. (Kinect for Windows, 2013).

Kinectin seuraava versio Kinect 2 ilmestyi marraskuussa 2013 Microsoftin Xbox One -konsolille. Seuraavan sukupolven Kinectin kameran tarkkuus on 1920x1080 pikseliä, joka on suuri harppaus vanhemman Kinectin VGA-kamerasta. Liikkeentunnistuskamera

ei ole enää lisälaite, vaan se tulee konsolin mukana ja on tärkeä osa konsolin kokonaisuutta.


KUVA 1. Kinect-tunnistuslaite

### 2.2.2 Asus Xtion PRO -laite

Tässä projektissa käytettiin liikkeentunnistukseen Asus Xtion PRO:ta. Xtionin kameran tarkkuus on Kinectin tapaan 640x480 pikseliä (Asus Xtion Pro tekniset tiedot). Myös alemman resoluution käyttö onnistuu, jolloin resoluutio laskee 320x240:een ja kuvataajuus kasvaa 30:stä 60:een. Asus Xtion PRO on Asuksen ja PrimeSensen yhteistyön tulos. Laite on suunniteltu nimenomaan OpenNI -ohjelmistokehystä silmälläpitäen.

KUVA 2. Asus Xtion PRO -kamera

# 3 PROJEKTIN SUUNNITTELU JA TOTEUTUS

Tässä luvussa kerrotaan Mind Reader –projektin lähtökohdista, etenemisestä, ongelmista ja lopputuloksesta.

## 3.1 Projektin lähtökohdat

Demola (Demola, 2013) on Tampereen Finlaysonilla sijaitseva innovaatiokeskus. Lukukausittain Demola kokoaa yhteen Tampereen korkeakoulujen opiskelijoita suorittamaan erilaisia projekteja. Projektit ovat yritysten tai korkeakoulujen sponsoroimia.

Mind Reader -projekti oli Tampereen ammattikorkeakoulun sponsoroima projekti Syksyllä 2012. Projektia toteuttamaan valittiin neljä opiskelijaa, mutta ryhmän koko karsiutui kolmeen yhden jätettyä projektin. Ryhmän kokoonpano oli taitojen kannalta graafikko, ohjelmoija sekä käyttökokemuksen- ja interaktiivisuuden osaaja.

KUVA 3: Ryhmän kokoonpano: Brianna Tsui, Lauri Nykänen ja Erika Kim.

### 3.2 Projektin tavoite ja tarkoitus

Projektin alkuperäisenä tavoitteena oli saada aikaan kaksi ohjelmaa: datansyöttöohjelma sekä peliohjelma, joka käyttäisi syötettyä dataa hyväksi. Projektin alkuperäisenä tarkoituksena oli selvittää web-kamerasta, minkälainen on ihmisen tunnetila ja tämän rinnalla selvittää liiketunnistuskamerasta käyttäjän liikkeet. Tunnetilaa oli tarkoitus käyttää projektissamme pelitoiminnallisuutta varten. Tunnetilan lisäksi haluttiin myös havainnoida liikkeitä käyttämällä liikkeentunnistuskameraa. Liikkeitä tunnistamalla pystyisimme kehittämään pelitoiminnallisuutta monimutkaisemmaksi, kuin pelkästään tunnetilaa seuraamalla.

### 3.3 Ohjelmointikielen valinta

Ohjelmiston toteutuskieleksi valittiin Java, sillä käyttöliittymä haluttiin tehdä HTML-pohjaisena. Tähän ratkaisuun vaikutti eniten projektiryhmän kokoonpano. Suurimmalla osalla ryhmästä ei ollut laajaa ohjelmointikokemusta, jonka vuoksi haluttiin käyttää mahdollisimman yksinkertaista ratkaisua käyttöliittymän suhteen, jotta osa ryhmästä pystyisi ylipäätänsä tekemään sitä. Projektin alkuvaiheissa pohdimme jopa Unityllä tehtyä käyttöliittymää, mutta totesimme, että projektiryhmän jäsenillä ei riitä aika ja taidot sen opettelemiseen.

### 3.4 Liiketunnistimen valinta

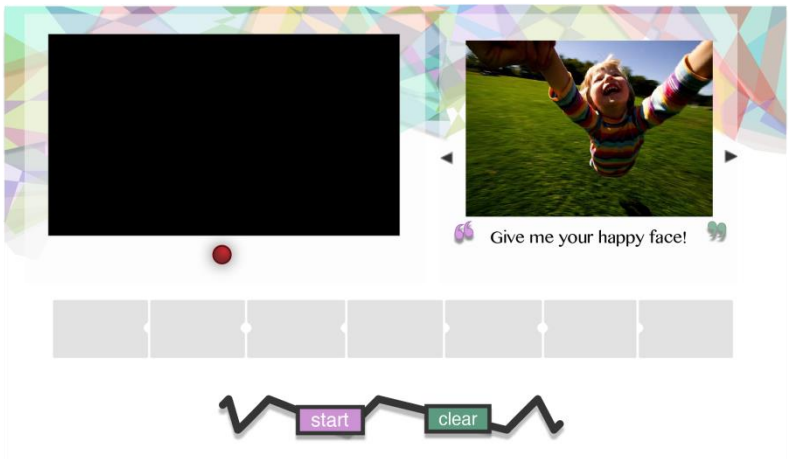
Vaihtoehdot projektissa käytettäväksi liiketunnistimeksi olivat Microsoft Kinect ja Asus Xtion. Kinectillä kehitys oli Xtioniin verrattuna rajallisempaa, joten valinta kohdistui Asus Xtioniin. Kinectin valitessamme olisimme joutuneet tekemään ohjelmiston C#:lla ja Microsoftin Visual Studio -kehitysympäristössä.

Halusimme kuitenkin toteuttaa ohjelmiston Javalla vapaammassa ympäristössä. Xtionin valitessamme saimme käyttöömme laajemmat mahdollisuudet ohjelmiston kehitykseen.

### 3.5 Pelin toimintojen suunnittelu

Pelin suunnitteluun annettiin tässä projektissa melko vapaat kädet. Pelin täytyi käyttää hyväksi web-kamerasta saatua tietoa pelaajan kasvojen ilmeestä. Sen piti myös olla helppo ymmärtää ja hauska pelata.

Päädyimme peliin, jossa käyttöliittymässä pelaajaa kehotetaan ilmeilemään tietty ilme ja pelaajalle annetaan pisteitä ilmeen suoritukseen menevän ajan perusteella. Onnistuneen ilmeen suorittamisen jälkeen käyttäjä näkee pistetilanteen muuttuneen ja ruudulla kehotetaan antamaan uusi, erilainen ilme. Kymmenen onnistuneen ilmeen jälkeen peli loppuu. Pelaajaa ilmeilyyn kannustamaan ilmekehotuksen rinnalle suunniteltiin näytettävän haluttuun ilmeeseen soveltuva kuva.




KUVA 4: Pelin käyttöliittymä suunnitteluasteella.

### 3.6 Työnjako

Ryhmä koostui graafisesta suunnittelijasta, käyttöliittymäasiantuntijasta ja ohjelmoijasta. Graafinen suunnittelija Erika Kim vastasi kaikesta ohjelman grafiikasta ja käyttöliittymän suunnittelusta. Kaksi muuta ryhmän jäsentä keskittyi ohjelman ohjelmistopuoleen. Minun vastuulleni annettiin ohjelmoida kehontunnistus kokonaisuudessaan. Tämän lisäksi jouduin ottamaan WWW-pohjaisen pelin ohjelmoinnin vastuulleni, kun tästä vastannut ryhmämme jäsen lähti projektista kesken kaiken pois. Käyttöliittymistä tietävä Bree Tsui joutui koodauspuolen hommiin, kun käyttöliittymässä ei loppujen lopuksi ollut paljoa suunniteltavaa. Hänen vastuulleen annettiin käyttäjän kasvojen ilmeiden tunnistuksen ohjelmointi.

Jokainen sitoutui työskentelemään 15 tuntia jokaisena viikkona projektin hyväksi. Tästä kulminoitui yhteensä 135 tuntia koko projektin aikana henkilöä kohden.

### 3.7 Aikataulutus

Projekti aloitettiin lokakuun puolessa välissä. Alussa keskityttiin lähinnä projektin esittelyyn tekemisen sijaan. Ensimmäinen toimiva versio ohjelmasta piti olla valmiina 5. joulukuuta. Ohjelman piti olla täysin valmis 4. tammikuuta ja valmista ohjelmaa oli tarkoitus esitellä 17. tammikuuta.

### 3.8 Ongelmat

Ensimmäiset ongelmat projektissa alkoivat, kun kasvojen tunnistuksen ohjelmoinnista vastannut Bree Tsui lähti marraskuun alussa kuukaudeksi Kiinaan. Yhteydenpitoa ei pidetty riittävästi hänen ja muun ryhmän välillä. Tämä johti siihen, että emme tienneet kuinka hänen työntekonsa sujui projektin parissa koko marraskuun ajan. Joulukuussa hänet tavatessamme kasvokkain meille selvisi että hänen osuutensa ei ollut edennyt yhtään hänen ollessa Kiinassa.

Suurempi ongelma ryhmällemme oli kuitenkin Jingjing Zhi, joka ei vastannut yhteydenottoihin eikä ilmestynyt kokouksiin projektin alun jälkeen. Joulukuun alussa jouduimme toteamaan, että emme voi enää pitää häntä projektin jäsenenä hiljaiselon vuoksi. Tämän seurauksena jouduimme tekemään radikaaleja leikkauksia vaatimusmäärittelyyn, jotta saisimme projektista toimivan demon aikarajaan mennessä valmiiksi.

Ryhmän ongelmat johtuivat pääosin siitä, ettei ryhmälle ollut nimettynä projektipäällikköä, joka olisi pitänyt projektin kasassa.

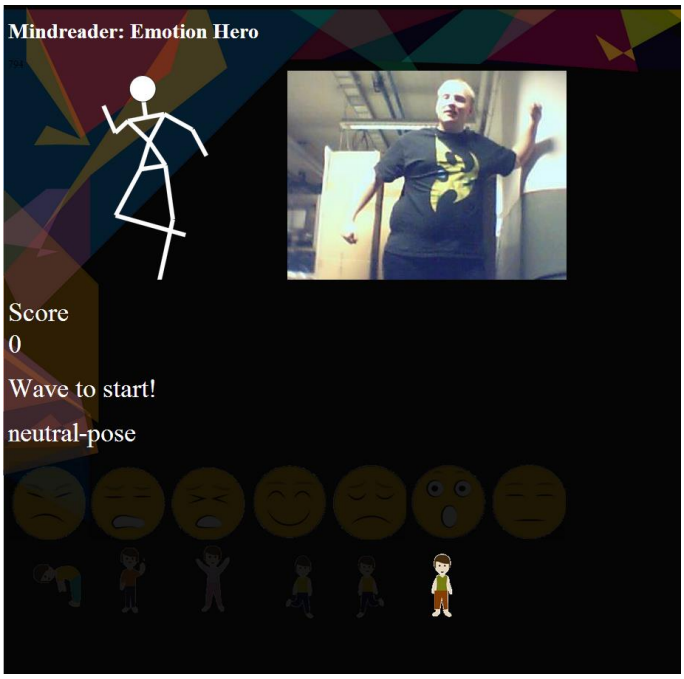
### 3.9 Toiminnallisuuksien karsiminen

Joulukuun alussa tuli selväksi, ettei kaikkia alkuperäiseen suunnitelmaan kuuluneita toiminnallisuuksia saada valmiiksi lopulliseen tuotteeseen. Demola-projekteissa suurin tavoite on saada aikaan demo, joten lähdimme tämä mielessä keskittymään toimivan demon aikaansaamiseen.

Ensimmäisenä luovuimme datansyöttöohjelmasta kokonaan. Aikaa tähän ei yksinkertaisesti riittänyt. Toisena mahdollisena kohteena oli kasvojen tunnistus. Tästä ei haluttu luopua, sillä koko projektin alkuperäisenä ajatuksena oli kasvojen tunnistus. Grafiikat ja HTML-käyttöliittymä olivat valmiit kasvojen tunnistusta varten, joten kasvojen tunnistamisen mukana pito ei sinänsä tuottanut muille ryhmän jäsenille lisätöitä.

### 3.10 Projektin lopputulos

Muutamaa viikkoa ennen projektin aikarajaa huomasimme, että kasvojen tunnistus ei valmistu aikarajaan mennessä, joten keskityimme vain liiketunnistuspuoleen. Myös datansyöttöohjelma peruuntui aikarajan takia. Lopputuloksena projektissa valmistui Kinectillä pelattava liiketunnistus-peli, jossa käyttöliittymä oli HTML-pohjainen ja itse peliä ajettiin Javalla.




KUVA 5: Lopullinen käyttöliittymä toiminnassa.

Lopullisessa pelissä käyttäjä seisoo noin kolmen metrin päässä Asus Xtion:sta ja nostaa kätensä ylös, jolloin laite osaa kalibroida käyttäjän sijainnin. Onnistunut kalibrointi näkyy käyttäjälle siten, että ruudun vasemmassa reunassa piirtyy kuva käyttäjän kehosta.

Peli aloitetaan nostamalla yksi käsi ylös. Tällöin käyttäjälle näytetään tavoiteltava kehonliike. Tavoiteltavan kehon liikkeen alapuolella pelissä näkyy tämänhetkinen koneen tulkitsema käyttäjän kehon liike. Tulkittu kehonliike näytetään myös pelin alareunassa kuvalla.

Käyttäjän tehdessä tavoiteltavan kehonliikkeen peli antaa käyttäjälle pisteitä tavoitellun liikkeen saavuttamiseen käytetyn ajan perusteella. Kymmenen liikkeen jälkeen peli loppuu ja käyttäjälle näytetään lopullinen pistelukema.

# 4 OHJELMISTON ARKKITEHTUURI

## 4.1 Ohjelmistoarkkitehtuurin määrittely

Mitä ohjelmistoarkkitehtuurilla tarkoitetaan ei välttämättä ole ihan helppo kysymys vastata. Tuntuu että jokaisella ihmisellä on hieman erilainen käsitys siitä, mitä ohjelmistoarkkitehtuuri on (Community software architecture definitions).

Ohjelmistoarkkitehtuurilla tarkoitetaan lyhyesti sanottuna projektin ratkaisujen kuvausta. Siinä kuvataan karkeasti järjestelmän komponentit ja voidaan sanoa, että se on summittainen kartta järjestelmästä (Defining architecture).

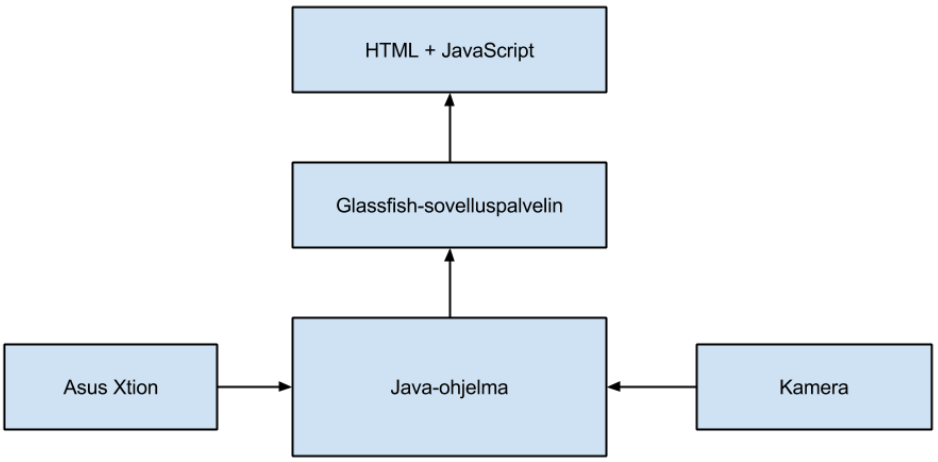
## 4.2 Ohjelmiston vaatimusmäärittely

Projektin alkuvaiheessa ryhmä muodostettiin ja keskustelimme siitä, minkälainen ohjelma on tavoitteena tehdä. Lähtökohtana oli tehdä interaktiivinen kasvojen- ja kehonliikkeen tunnistuspeli ja sen rinnalle datansyöttöpeli, jota voitaisiin käyttää pelissä hyödyksi.

Ohjelmamme pitäisi tunnistaa seitsemän erilaista ilmettä: iloisen, surullisen, neutraalin, vihaisen, yllättyneen, vastenmielisen ja pelokkaan. Ohjelman käytön pitää olla helppoa ymmärtää ja hauska käyttää. Ohjelman ulkoasun ei tarvitse olla hienon näköinen; tärkeintä on, että logiikka toimii. Pelin ei ole tarkoitus olla monimutkainen, vaan se tehdään vain toimiakseen teknologiademona.

Projektin edetessä myös vaatimusmäärittely muuttui. Joulukuussa projekti ei ollut edennyt odotetunlaisesti. Tämän takia jouduimme jättämään datansyöttöpelin lopullisesta tuotteesta pois. Tämän lisäksi kasvojentunnistuksen sijaan jouduimme tunnistamaan vain kehon liikkeitä, sillä ohjelman kasvojentunnistusosa ei valmistunut aikataulun puitteissa. Näitä kahta suurta kokonaisuutta lukuunottamatta lopullinen tuote vastasi alkuperäistä vaatimusmäärittelyä.

### 4.3 Karkean tason toiminnallisuus




KUVA 6: Ohjelman toiminta karkeasti esitettynä.

Ohjelmakokonaisuus koostuu viidestä isosta osasta. Asus Xtion –kamerasta saadaan kuva, jota analysoidaan Java-ohjelmassa. Web-kameraa oli tarkoitus alunperin käyttää kasvojen ilmeiden tunnistamiseen, mutta lopullisessa versiossa sen tarkoitus on antaa käyttäjälle peilikuva itsestään, jota hän voi verrata Java-ohjelman generoimaan SVG-kuvaan. Java-ohjelma selvittää käyttäjän kehon liikkeen ja tallentaa tiedon paikallisesti tekstitiedostona. Glassfish-sovelluspalvelin välittää Java-ohjelman tallentamat kuvat ja tiedon käyttäjän tekemästä liikkeestä käyttöliittymänä toimivalle HTML- ja JavaScript -kokonaisuudelle.

### 4.4 Java-ohjelma

Java-ohjelma on tämän projektin tärkein osa. Ohjelma tunnistaa käyttäjän kehon ja tallentaa siitä kehon liikkeen, jonka Glassfish-sovelluspalvelin välittää eteenpäin käyttöliittymälle. Ohjelma rakennettiin OpenNI:n esimerkkisovelluksen päälle. Java-ohjelma suorittaa liikkeentunnistuksen käyttämällä OpenNI-kehyksen NITE-moduulia hyödyksi. Ohjelma tulkitsee käyttäjän kehon raajojen pisteet kolmiulotteisessa avaruudessa ja tallentaa pisteistä tulkitun liikkeen tekstitiedostona.

### 4.4.1 OpenCV- ja OpenNI-kehys

OpenNI on ohjelmointirajapintojen kokoelma liikkeentunnistusta varten . Sen yhtenä osana on PrimeSensen NITE-moduuli, joka suorittaa tässä Java-ohjelmassa liikkeentunnistuksen. Java-ohjelmasta saadaan ulos video Asus Xtion –kameran näkymästä käyttämällä OpenCV-kirjastoa.

### 4.4.2 Funktiot

Java-ohjelman keskiössä on paint-funktio, joka pyörii jatkuvasti ja kutsuu muita funktioita. DrawSkeleton-funktio piirtää käyttäjän ruumiin Java-ohjelman omaan JFrameen ja sen jälkeen kutsuu datapisteiden tulkitsemiseen käytettäviä funktioita. Datapisteiden tulkintaa hoitaa kolme eri funktiota; käsien, jalkojen ja pään liikkeille on omat funktionsa.

### 4.4.3 Datapisteiden tulkitseminen

Käyttäjän kehon datapisteistä liikkeen tulkinta on oleellinen osa Java-ohjelman toimintaa. Tulkittavia liikkeitä olivat käsien ja jalkojen liikutus, sekä kumarrus. Käsistä haluttiin tietää olivatko ne ylhäällä ja jos oli niin oliko vain jompikumpi käsi ylhäällä. Käyttäjän jaloista katsottiin liikuttaminen syvyys- ja leveyssuunnassa. Vasen ja oikea jalka eriteltiin toisistaan.

Käsien liikkeen tunnistusta hoitaa areHandsUp-funktio (Liite 1). Funktiossa katsotaan hetkelliset käyttäjän pään ja käsien paikat ja näiden perusteella tehdään tulkinta siitä, että onko jompikumpi- tai kummatkin kädet ylhäällä. Pään ja käsien etäisyys toisistaan lasketaan ja etäisyyden ollessa pieni todetaan käden olevan ylhäällä. Käsien ollessa ylhäällä tekstitiedostoon tallennetaan tästä tieto.

Jalkojen liikettä tarkastellessa ei voitu käyttää samanlaista kiintopistettä kuten käsien laskennassa käytettiin päätä, vaan laskennassa käytettiin jalkojen paikkaa ajan suhteen. Funktiossa legs laskettiin kummankin jalan hetkellinen sijainti ja tätä verrattiin viime kerran laskettuun sijaintiin (Liite 2). Tarpeeksi suuri ero tulkittiin jalan liikkeeksi.

Kumarrusta tulkittaessa käytettiin vain käyttäjän pään sijaintia ja aivan kuten jaloissa sen sijantia tarkasteltiin ajan suhteen (Liite 3). Tässä otettiin vielä huomioon vain pään alaspäin suuntautunut liikesuunta, eli vain alaspäin liikkuva pää tulkittiin kumarrukseksi. Mikäli minkään liikkeen ei todettu tapahtuneen, tulkittiin käyttäjän olemus neutraaliksi.

### 4.4.4 Liikkeen tunnistus

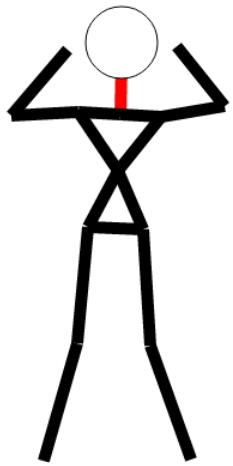
Ohjelmassa liikkeen tunnistukseen käytetään OpenNI-kehyksen NITE-moduulia, joka tarjoaa valmiit funktiot liikkeen tunnistukseen. Otimme java-ohjelmamme pohjaksi OpenNI:n esimerkkiohjelman, joka tunnisti käyttäjän kameran kautta. Ohjelma yrittää kalibroida kameran kuvasta käyttäjää, kunnes se onnistuu siinä. Käyttäjä joutuu nostamaan kätensä ylös, jotta käyttäjän kehon pisteiden kalibrointi onnistuisi. Kalibroinnin suoritettuaan ohjelma seuraa käyttäjän kehoa.

Ohjelma tukee myös useamman henkilön samanaikaista tunnistusta, mutta rakensimme ohjelman vain yksi käyttäjä mielessä, joten monen käyttäjän samanaikainen pelin pelaaminen ei käytännössä onnistu. Ohjelma tunnistaa käyttäjästä viisitoista kehon eri pistettä kolmiulotteisessa avaruudessa.

### 4.4.5 SVG-kuvan luominen ja tallennus

SVG-kuva on World Wide Web Consortiumin kehittämä kaksiulotteisten kuvien kuvauskieli (SVG, 2013). SVG on lyhenne sanoista Scalable Vector Graphics. Tässä projektissa SVG-kuva muodostetaan liikkeentunnistuskamerasta saatujen käyttäjän kehon datapisteistä. Kuva muodostetaan käyttämällä hyödyksi Apache Batik –kirjastoa (Batik, 2013). SVG-kuvan muodostamisen jälkeen kuva muutetaan PNG-kuvaksi, sillä Glassfish-sovelluspalvelin ei pystynyt käyttämään suoraan SVG-kuvaa. JavaScript-skripti pyytää sovelluspalvelimelta PNG-kuvan piirtääkseen sen käyttöliittymään.

Java-ohjelmassa funktio saveSVG rakentaa SVG-kuvan käyttämällä OpenNI-kirjaston SkeletonJointista saatuja käyttäjän ruumiinosia vastaavia datapisteitä. Funktio svgToPng puolestaan muodostaa PNG-kuvan juuri tehdystä SVG-kuvasta.



KUVA 7: Generoitu SVG-kuva.

### 4.4.6 Datapisteiden tallennus tietokantaan

Ohjelmaan tehtiin runko datapisteiden tallennuksesta tietokantaan. Valmiissa ohjelmassa näitä funktioita ei kuitenkaan käytetä, sillä opetusosuudesta luovuttiin, jotta saisimme toimivan demon aikataulun puitteissa valmiiksi. Tietokanta on HSQLDB-tietokanta ja siihen pystyy tallentamaan käyttäjän kehon pisteet kolmiuloitteisessa avaruudessa.

## 4.5 Käyttöliittymä

Käyttöliittymä tehtiin HTML:llä ja siihen liittyvä toiminta toteutettiin JavaScript-skripteillä. Käyttöliittymä pyytää Glassfish-sovelluspalvelimelta kehosta piirretyn kuvan, tiedon käyttäjän kehon tämänhetkisestä liikkeestä sekä kuvan web-kamerasta.

Käyttöliittymän skripteistä tärkein on widget_emo_viewer. Sen funktio getImg piirtää web-kamerasta tallennetun kuvan pelitilantesta. Funktio changeOpacity häivyttää muiden liikeiden ikonit paitsi sen, jota käyttäjä Java-ohjelman tulkinnan mukaan tällä hetkellä tekee. Funktio randomDoThis sisällyttää kokonaan yksinkertaisen pelitoiminnallisuuden toteutuksen. Funktio getEmotion pyytää Glassfish-sovelluspalvelimelta Java-ohjelman tekstitiedostona tallentaman tiedon käyttäjän liikkeestä.

### 4.6 Glassfish-sovelluspalvelin

Glassfish-sovelluspalvelin oli tarpeellinen, sillä käyttöliittymämme ei voinut lukea suoraan Java-ohjelman tallentamia tiedostoja, joissa oli tieto käyttäjän kehon liikkeestä sekä generoitu kuva käyttäjästä. Java-ohjelman paikallisesti tallentamien tietojen saaminen käyttöliittymälle oli tämän projektin suurin yksittäinen haaste.

Palvelin vastaa getEmotion-kyselyyn palauttamalla tekstinä käyttäjän liikkeen, esimerkiksi käyttäjän liikkeen ollessa kummarus palautettu teksi on muotoa ”bow”. PNG-kuvaksi muunnettu SVG-kuva palautetaan vastaamalla kyselyyn svgpng. Web-kamerasta tallennettu kuva saadaan palvelimesta webcam-kyselyllä.

### 4.7 Web-kamera

Projektin alkuperäisenä tavotteessa oli kasvoista ilmeen tunnistus. Tätä toimintoa varten ohjelmaan tehtiin toiminto web-kameran kuvan tallentamista varten. Kasvojen tunnistuksen teosta luopumisen jälkeen web-kameran kuva päätettiin kuitenkin pitää ohjelmassa mukana.

Lopullisessa ohjelmassa web-kamerasta otetaan kuvia 10 millisekunnin välein ja kuva välitetään käyttöliittymälle Glassfish-sovelluspalvelimen kautta. Ohjelmassa web-kameran kuva otetaan käyttämällä JavaCV:n FrameGrabber –luokkaa. Web-kameran kuva näytetään generoidun SVG-kuvan kanssa käyttöliittymässä rinnakkain, jolloin käyttäjä voi verrata todellista kuvaa Java-ohjelman luomaan kuvaan.

# 5 YHTEENVETO

Projektin alkuperäisenä tavoitteena oli tehdä kaksi ohjelmaa liiketunnistuksen teknologiademoksi. Toinen datansyöttöön ja toinen tätä hyödyntävä teknologiademo. Projektipäällikön puuttumisesta seurannut projektiryhmäläisten toimettomuus ja yhden jäsenen projektista lähteminen johti näistä tavoitteista luisumiseen. Projektin tavoitteita jouduttiin katsomaan uudestaan projektin puolessavälissä, jolloin suurin osa ominaisuuksista leikattiin. Leikkauksen kohteeksi päätyi datansyöttöohjelma ja sitä hyödyntävä kasvojentunnistuspeli. Jälkikäteen katsoen ominaisuuksien leikkaus oli oikea ratkaisu, sillä muuten emme varmasti olisi saaneet valmista demoa aikaan aikarajaan mennessä.

Lopputuloksena valmistunut liikkeentunnistuspeli tunnistaa käyttäjästä kuusi erilaista liikettä. Lopullinen ohjelma toimii lähes moiteettomasti, mutta se vaatii paljon alkuvalmisteluita käynnistyäkseen. Harmittavasti projektissa aika loppui kesken, sillä olimme hyvin lähellä saada katseentunnistuksen mukaan lopulliseen peliin.

Ohjelman ohjelmistoarkkitehtuurista muodostui mielenkiintoinen kokonaisuus. Java-ohjelmaa käytetään liikkeentunnistukseen ja siitä saatujen tietojen tulkitsemiseen ja tallentamiseen. Käyttöliitymän tekeminen HTML-pohjaisena ei lopulta ehkä ollutkaan paras ratkaisu, sillä sen käyttäminen johti yllättäviin ongelmiin, erityisesti Java-ohjelman tallentamien tietojen luvun kanssa. Suurin vaikeus projektissa oli saada Java-ohjelman paikallisesti tallentamat tiedot käyttöliittymälle. Tätä tarkoitusta varten projektissa jouduttiin turvautumaan Glassfish-sovelluspalvelimeen, joka välittää tiedon liikkeestä ja kuvista käyttöliittymälle.

Minulle suurin oppi tässä projektissa oli projektipäällikön tärkeys. Projektipäällikön puuttuminen tässä projektissa aiheutti suuria ongelmia, jotka johtivat tavoitteista poikkeavaan lopputulokseen. Projektiryhmän jäsenet voivat kuitenkin olla tyytyväisiä, että ongelmista huolimatta projektista saatiin valmiiksi toimiva teknologiademo.

**LÄHTEET**

Asus Xtion Pro tekniset tiedot. Luettu 14.5.2013.
http://www.asus.com/Multimedia/Xtion_PRO/#specifications

Batik. Luettu 15.11.2013.
http://xmlgraphics.apache.org/batik/

Community software architecture definitions. Luettu 15.5.2013.
http://www.sei.cmu.edu/architecture/start/glossary/community.cfm

Defining architecture. Luettu 15.5.2013.
http://www.iso-architecture.org/ieee-1471/defining-architecture.html

Demola kotisivu. 2013. Luettu 8.11.2013.
http://tampere.demola.fi/

Kinect for Windows. Luettu 20.11.2013.
http://www.microsoft.com/en-us/kinectforwindows/discover/features.aspx

Melgar, E. Díez, C. 2012. Arduino and Kinect Projects: Design, Build, Blow Their Minds. New York: Apress.
http://proquest.safaribooksonline.com.elib.tamk.fi/book/-/9781430241676

PrimeSense NITE. Luettu 20.11.2013.
http://www.primesense.com/solutions/nite-middleware/

SVG. Luettu 15.11.2013.
http://www.w3.org/Graphics/SVG/

# LIITTEET

## Liite 1. AreHandsUp-funktio

```
public void areHandsUp(int user)throws StatusException {
    getJoints(user);
    HashMap<SkeletonJoint, SkeletonJointPosition> dict = joints.get(new Integer(user));

    Point3D pos_head= dict.get(SkeletonJoint.HEAD).getPosition();
    int hpX = (int) pos_head.getX();
    int hpY = (int) pos_head.getY();
    int hpZ = (int) pos_head.getZ();

    Point3D pos_righHand= dict.get(SkeletonJoint.RIGHT_HAND).getPosition();
    int rhX = (int) pos_righHand.getX();
    int rhY = (int) pos_righHand.getY();
    int rhZ = (int) pos_righHand.getZ();

    Point3D pos_leftHand= dict.get(SkeletonJoint.LEFT_HAND).getPosition();
    int lhX = (int) pos_leftHand.getX();
    int lhY = (int) pos_leftHand.getY();
    int lhZ = (int) pos_leftHand.getZ();

    double d = Math.sqrt(Math.pow((hpX-rhX), 2)+Math.pow((hpY-rhY), 2)+Math.pow((hpZ-rhZ), 2));
    double d2 = Math.sqrt(Math.pow((hpX-lhX), 2)+Math.pow((hpY-lhY), 2)+Math.pow((hpZ-lhZ), 2));

    if ( d < 200 || d2 < 200) {
        File filename = new File("emo.txt");
        PrintWriter writer;
        try {
            currentEmotion = "'wave'";
            writer = new PrintWriter(new FileOutputStream(filename));
            writer.println("wave");
            writer.flush();
            writer.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
            userHandsUp = true;
    }
    else {
        neutral = true;
        File filename = new File("emo.txt");
        PrintWriter writer;
        try {
            currentEmotion = "'neutral'";
            writer = new PrintWriter(new FileOutputStream(filename));
            writer.println("neutral-pose");
            writer.flush();
            writer.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        userHandsUp = false;
    }

     if ( d < 200 && d2 < 200) {
         File filename = new File("emo.txt");
         PrintWriter writer;
         try {
             currentEmotion = "'both-hands-up'";
             writer = new PrintWriter(new FileOutputStream(filename));
             writer.println("both-hands-up");
             writer.flush();
             writer.close();
         } catch (FileNotFoundException e) {
             e.printStackTrace();
         }
         userHandsUp = true;
     }

     if( userHandsUp == true ) {
         neutral = false;
     }
 }
```

# Liite 2. Legs-funktio

```
public void legs(int user)throws StatusException {
    footCounter++;
    getJoints(user);
    HashMap<SkeletonJoint, SkeletonJointPosition> dict = joints.get(new Integer(user));
    Point3D pos_rfoot= dict.get(SkeletonJoint.RIGHT_FOOT).getPosition();
    int fpX = (int) pos_rfoot.getX();
    int fpY = (int) pos_rfoot.getY();
    int fpZ = (int) pos_rfoot.getZ();

    if (footCounter == 20) {
        lastRFoot[0] = fpX;
        lastRFoot[1] = fpY;
        lastRFoot[2] = fpZ;
    }

    double df = Math.sqrt(Math.pow((fpX-lastRFoot[0]), 2)+Math.pow((fpY-lastRFoot[1]), 2)+Math.pow((fpZ-lastRFoot[2]), 2));

    if ( df > 50) {
        File filename = new File("emo.txt");
        PrintWriter writer;
        try {
            currentEmotion = "'right-foot-moving'";
            writer = new PrintWriter(new FileOutputStream(filename));
            writer.println("right-foot-moving");
            writer.flush();
            writer.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }

    Point3D pos_lfoot= dict.get(SkeletonJoint.LEFT_FOOT).getPosition();
     int lfpX = (int) pos_lfoot.getX();
     int lfpY = (int) pos_lfoot.getY();
     int lfpZ = (int) pos_lfoot.getZ();

     if (footCounter == 20) {
        lastLFoot[0] = lfpX;
        lastLFoot[1] = lfpY;
        lastLFoot[2] = lfpZ;

        footCounter = 0;
     }

     double ldf = Math.sqrt(Math.pow((lfpX-lastLFoot[0]), 2)+Math.pow((lfpY-lastLFoot[1]), 2)+Math.pow((lfpZ-lastLFoot[2]), 2));

     if ( ldf > 50) {
        File filename = new File("emo.txt");
        PrintWriter writer;
        try {
            currentEmotion = "'left-foot-moving'";
            writer = new PrintWriter(new FileOutputStream(filename));
            writer.println("left-foot-moving");
            writer.flush();
            writer.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
     }
}
```

Liite 3. Bow-funktio

```
public void bow(int user)throws StatusException {
    bowCounter++;
    getJoints(user);
    HashMap<SkeletonJoint, SkeletonJointPosition> dict = joints.get(new Integer(user));

    Point3D pos_head= dict.get(SkeletonJoint.HEAD).getPosition();
    int lHz = (int) pos_head.getZ();

    if (bowCounter == 5) {
        lastHead = lHz;
        bowCounter = 0;
    }

    double headMovement = lHz-lastHead;

    if ( headMovement < -200) {
        currentEmotion = "'bow'";
        changeMotionCounter = 0;
        neutral = false;
        File filename = new File("emo.txt");
        PrintWriter writer;
        try {
            writer = new PrintWriter(new FileOutputStream(filename));
            writer.println("bow");
            writer.flush();
            writer.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
    else {
        neutral = true;
    }
}
```