Toivo Honkanen

OpenNI-ohjelmistokehitysympäristö liiketunnistuksessa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

23.3.2014

Metropolia

Tekijä Otsikko Sivumäärä Aika	Toivo Honkanen OpenNI-ohjelmistokehitysympäristö liiketunnistuksessa 26 sivua + 1 liite 23.3.2014
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	ohjelmistotekniikka
Ohjaaja	yliopettaja Jarkko Vuori

Työn tavoitteena oli tutkia OpenNI 2.x ohjelmointikehitysympäristön ominaisuuksia ja sen päälle rakennettuja ohjelmistoja sekä Asuksen ja PrimeSensen yhdessä kehittämän liiketunnistinohjaimen käyttöä kotitietokoneen ohjauksessa.

Liiketunnistinohjaimen ominaisuuksia tutkittiin Asus Xtion-ohjaimen ja sen mukana toimitettujen ohjelmistojen avulla. Lisäksi sekä ohjaimen että OpenNI:n ominaisuuksien tutkimiseksi tehtiin OpenNI-ohjelmointiympäristön työkaluja käyttäen esimerkkisovellus, joka on yksinkertainen piirto-ohjelma, jolla käyttäjä voi muutamaa eri väriä käyttäen piirtää kuvan pelkällä kädellä. Sovellus toteutettiin Java-ohjelmointikielellä.

Ohjain vaikutti olevan melko huono hiiren tai kosketusnäytön korvaaja kotitietokoneen ohjaimena epätarkkuutensa ja tilantarpeensa vuoksi.

Sovellus toimi, mutta sillä ei ole juurikaan todellista käyttöä, koska piirtäminen ei ole kovin tarkkaa. Sovelluksen tekeminen antoi kuitenkin hyvän kuvan siitä kuinka ohjainta voidaan hallita OpenNI:n ohjelmointityökaluilla.

OpenNI on hyvin matalan tason järjestelmä, jolla päästään käsiksi vain liiketunnistinohjaimen antamaan raakaan dataan. Mikäli halutaan tehdä vähänkin monimutkaisempia sovelluksia, kannattaa hyödyntää jonkun muun tekemiä korkeamman tason työkaluja.

Avainsanat	OpenNI, liikkeentunnistus, Xtion

Author	Toivo Honkanen
Title	Using an OpenNI software developing kit with a motion sensor
Number of Pages	26 pages + 1 appendix
Date	23 March 2014
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor	Jarkko Vuori, Principal Lecturer

The goal of the thesis was to study the characteristics of the OpenNI 2.x software development kit and software developed for it. Another purpose was to study a motion sensor developed by Asus and PrimeSense to see how it works as the controller of a personal computer.

The characteristics of the sensor were studied with Asus Xtion and its software. To study the sensor and the SDK an application was created using OpenNI developing tools. The developed application is a simple drawing program with which the user can create a drawing with bare hands. The application was created with the Java programming language.

The sensor seemed not to be very good for controlling a personal computer due to its inaccuracy. Using the controller also needs a lot of space.

The application worked well but it does not have any real use because drawing with it is too inaccurate. However, making the application gave a good idea about how to control the motion sensor with the OpenNI software tools.

In conclusion, the OpenNI SDK is a low level system. One can only get raw data from the motion sensor. If there is need for developing a more complex application, the use of middleware libraries is highly recommended.

Keywords	OpenNI, motion tracking, Xtion

Sisällys

Liite 1. Sovelluksen lähdekoodi

Lyhenteet

3D	*Three dimensional.* Kolmiulotteinen tai grafiikka, joka on mallinnettu kolmen tilaulottuvuuden suhteen.
API	*Application programming interface.* Ohjelmointirajapinta.
EEPROM	*Electrically Erasable Programmable Read-Only Memory.* Sähköisesti tyhjennettävissä ja ohjelmoitavissa oleva lukumuisti.
HTML	*Hypertext Markup Language.* Kuvauskieli, jolla voidaan kuvata hyperlinkkejä sisältävää tekstiä.
RGB	*Red, green, blue.* Väriavaruus, jossa eri värit muodostetaan sekoittamalla punaista, vihreää ja sinistä valoa.
SDK	*Software development kit.* Ohjelmointikehitysympäristö.
URI	*Uniform Resource Identifier.* Merkkijono, jolla kerrotaan tietyn tiedon paikka tai yksikäsitteinen nimi.
USB	*Universal Serial Bus.* Sarjaväyläarkkitehtuuri oheislaitteiden liittämiseksi tietokoneeseen.

1 Johdanto

Microsoft julkaisi marraskuussa 2010 Xbox 360-pelikonsolilleen aivan uudentyyppisen ohjaimen, jossa ei ole ollenkaan kädessä pidettävää peliohjainta. Tämä nimen Kinect saanut ohjain perustuu sen sijaan pelaajan liikkeet tunnistaviin sensoreihin. Tästä uudesta ohjaimesta tuli välittömästi suuri myyntimenestys. Ohjain perustuu PrimeSense-yhtiön alun perin kehittämään liikkeentunnistustekniikkaan. [1.]

Jo saman kuukauden aikana ilmestyi ensimmäinen avoimen koodin ajuri Kinect-ohjaimen liittämiseksi kotitietokoneisiin. Microsoft vastusti tätä kehitystä aluksi jyrkästi, mutta jo seuraavan vuoden helmikuussa yhtiö ilmoitti julkaisevansa seuraavana syksynä täysin ilmaisen ohjelmistokehitysympäristön Windowsille Kinetct-ohjainta varten. [2; 3.]

Marraskuussa 2010 perustettiin myös OpenNI, joka on voittoa tavoittelematon yhteenliittymä. Yhteenliittymän tarkoituksena on edistää luonnollisten käyttöliittymien (engl. natural interface) laitteistojen ja ohjelmistojen yhteensopivuutta ja yhteistoimintaa. OpenNI-yhteisö tarjoaa ohjelmistojen kehittäjille avoimen koodin ohjelmointityökalut, joilla voidaan kehittää ohjelmistoja kaikille yhteensopiville liiketunnistinohjaimille. PrimeSense Ltd on tämän yhteenliittymän merkittävin toimija ja perustajajäsen. Vuonna 2012 PrimeSense ja Asus kehittivät yhteistyössä uuden Kinect-ohjaimen kaltaisen liiketunnistinohjaimen, joka on tarkoitettu pelikonsolien sijaan käytettäväksi tavallisissa kotitietokoneissa. Tätä OpenNI-yhteensopivaa laitetta myydään sekä Asus Xtion että PrimeSense Sensor -tuotenimillä. [4; 5; 6.]

Insinöörityön tarkoituksena on tutkia OpenNI-ohjelmointiympäristön mahdollisuuksia sekä Asus Xtionin ominaisuuksia kotitietokoneen ohjaimena. Tavoitteena on myös kehitellä sovellus, jossa hyödynnetään OpenNI-ohjelmointirajapintaa.

2 Liikkeen tunnistavat ohjaimet

Liikkeen tunnistavissa ohjaimissa on oltava jotain, joka liikkuu, anturit, jotka tunnistavat liikkeitä, sekä ohjelmisto, joka muuttaa anturien havaitsemat liikkeet tietokoneen tai pelikonsoliin ymmärtämäksi dataksi. Toteutustavat tällaisille ohjaimille ovat nykyään

lähinnä kameraan perustuva tunnistus, kiihtyvyysantureihin perustuva tunnistus tai näiden yhdistelmä. [7.]

2.1 Luonnolliset käyttöliittymät

Tietokonetekniikassa luonnollisella käyttöliittymällä (engl. natural user interface) tarkoitetaan käyttöliittymää, joka on näkymätön tai käyttäjä kokee sen näkymättömäksi. Sanaa luonnollinen käytetään, koska siinä ei käytetä mitään keinotekoista laitetta, kuten hiirtä tai näppäimistöä. Luonnollisen käyttöliittymän käytön oppimisen tulisi olla helppoa ja sen pitäisi perustua ihmisen luontaiselle käyttäytymiselle. Esimerkkinä tällaisesta käyttöliittymästä on vaatteisiin integroitu ohjain. Käyttäjä voi sellaista käyttäessään esimerkiksi tarttua virtuaaliseen esineeseen, aivan kuten hän tarttuisi oikeaan esineeseen.

Liiketunnistinohjaimella toteutettua käyttöliittymää pidetään luonnollisena käyttöliittymänä, koska siinä ei ole fyysistä kosketusta mihinkään laitteeseen vaan se perustuu käyttäjän liikkeitä ja paikkatietoja mittaaviin sensoreihin. [8.]

2.2 OpenNI-yhteensopivat ohjaimet

Vaikka OpenNI-yhteenliittymä kertookin kehittävänsä luonnollisten käyttöliittymien ohjelmistoja, se tosiasiassa keskittyy ainoastaan liiketunnistinohjaimiin. Tällöin yhteensopivia laitteita ovat käytännössä ainoastaan PrimeSense-yhtön kehittämään liikkeentunnistusteknologiaan perustuvat laitteet. [8.]

OpenNI-yhteensopivien ohjainten tekniikkaa on esitelty luvussa viisi, jossa kerrotaan tarkemmin Asus Xtion-ohjaimesta.

2.3 Kilpaillevat ohjaimet

Erilaisia pelaajan liikkeet tunnistavia ohjaimia on kehitetty jo 1990-luvun alkupuolelta lähtien. Ensimmäisistä tällaisista ohjaimista ei kuitenkaan koskaan tullut kaupallista menestystä. Varsinainen läpimurto liikkeet tunnistavissa ohjaimissa tapahtui vasta vuonna 2006 Nintendon julkaistua Wiin ja sen Wii Remote -ohjaimen. Tämän jälkeen

Sony kehitti PlayStation Moven ja Microsoft Kinect-ohjaimen ratkaisuikisiksi pelien ohjaukseen liikkeiden avulla. [9.]

Wii Remote

Wii Remotessa, joka tunnetaan myös nimellä Wiimote, liikkeiden tunnistus tapahtuu kiihtyvyysanturien avulla. Siinä on lisäksi optinen sensori, jota voidaan käyttää osoitinlaitteena ja kahdeksan toimintonäppäintä sekä muista peliohjaimista tuttu neljään suuntaan painettava d-padiksi kutsuttu näppäin, joka kuvassa 1 näkyy ylimpänä. Kommunikaatio pelikonsolin ja ohjaimen välillä tapahtuu Bluetooth-yhteydellä. Ohjaimessa on oma kaiutin ja pieni EEPROM-muistipiiri. Voimanlähteekseen ohjain tarvitsee kaksi AA-kokoista paristoa. [10.]

Kuva 1. Wii Remote. [9.]

PlayStation Move

Jo vuonna 2003 Sony julkaisi peliohjaimen nimeltä EyeToy ja sen seuraajan PlayStationEyen. Nämä perustuvat hahmon tunnistukseen RGB-kameralla ja mikrofoniin. Näillä ohjaimilla ei kuitenkaan voi pelata muita kuin niille varta vasten sille suunniteltuja pelejä. [11; 12.]

Vuonna 2010 Sony julkaisi PlayStation Moven, joka on kädessä pidettävä sauva ja toimii yhdessä PlayStationEyen kanssa. Liiketunnistus perustuu siihen, että sauvan yläpäässä on valaistu pallo, jonka liikkeitä kamera kuvaa. Ohjain palloineen on esitetty kuvassa 2. Etäisyys lasketaan kameran kuvasta pallon koon perusteella. Ohjaimessa on myös antureita, jotka tunnistavat liikkeitä ja sauvan asennon. Asennon kalibrointi tapahtuu Maan magneettikentän avulla. Lisäksi ohjaimessa on toimintonäppäimiä. Kommunikointi pelikonsolin kanssa tapahtuu Bluetooth-yhteydellä, ja voimanlähteenä on USB-liitännän kautta ladattava litiumakku. [12.]

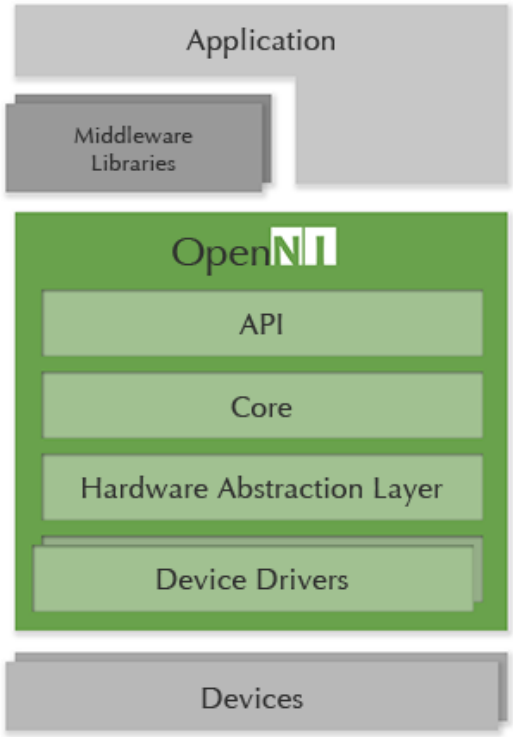
Kuva 2. PlayStation Move. [12.]

3 OpenNI 2.0 SDK

3.1 Ohjelmointiympäristön rakenne

OpenNI SDK on ohjelmistokehitysympäristö (engl. Software Development Kit). Se perustuu avoimeen lähdekoodiin ja on kehitetty 3D-liiketunnistinohjaimia varten. Sitä pidetään kyseisten laitteiden ohjelmistokehityksen standardina ja Openni-yhteisö on suurin 3D-liikkeentunnistuksessa käytettäviä ohjelmistoja kehittävä yhteisö. OpenNI SDK:ta ladataan yli 85 tuhatta kertaa kuukausittain. [6.]

OpenNI SDK 2 tarjoaa ohjelmistojen kehittäjille alustan väliohjelmakirjastojen (engl. middleware libraries) ja sovellusten kehittämiseksi liikkeentunnistukseen perustuville ohjainlaitteille. Alusta koostuu laiteajureista, jotka ovat suorassa yhteydessä liikettä tunnistaviin laitteisiin laitteiden abstraktiokerroksesta, ytimestä ja ohjelmointirajapinnasta. OpenNI:n rakenne on kuvassa 3, jossa varsinainen SDK on merkitty vihreällä. SDK:n päälle kehittäjät voivat rakentaa väliohjelmistoja (engl. middleware), joita käytetään työkaluina sovellusten kehityksessä sekä SDK:ta hyödyntäviä sovelluksia. SDK tukee uusimman sukupolven entistä lyhyemmältä etäisyydeltä toimivia sensoreitta. [6.]



Kuva 3. OpenNI:n rakenne. [5.]

OpenNI SDK:n oikeuksien haltija on PrimeSense Ltd, ja se on lisensoitu Apache 2.0 lisenssillä. Lisenssi sallii ohjelmistojen vapaan levityksen ja muokkaamisen, kunhan ne julkaistaan saman lisenssin alaisuudessa ja kaikista muokkauksista lisätään huomautus. Ohjelmistoilla ei lisenssin perusteella myöskään ole minkäänlaista takuuta. [13; 14.]

Tuetut käyttöjärjestelmät ovat OpenNI 2.2 Beta-julkaisusta alkaen Windows XP SP2 tai uudempi, Windows 7, Windows 8 ja Ubuntu 12.04 Linux tai uudempi. Kaikista näistä on omat versionsa sekä 32-bittisille että 64-bittisille käyttöjärjestelmille. Lisäksi tuetaan Applen OSX 10.7:ää tai uudempaa käyttöjärjestelmää. [15; 16.]

3.2 Ohjelmointirajapinta eli API

API (engl. Application Programming Interface) tarjoaa yhden yhtenäistetyn ohjelmointirajapinnan OpenNI-yhteensopiville liiketunnistinohjaimille. Sen avulla voidaan alustaa sensorit ja vastaanottaa videovirtaa syvyys-, RGB- ja infrapunakameroilta. Videovirran lukeminen voi perustua joko silmukkaan tai tapahtumapohjaiseen lukemiseen. Silmukkaan perustavassa tavassa kutsutaan kuvakehyksen lukevaa metodia, joka jää odottamaan, kunnes kehys on valmis ja lukee sen sitten. Tapahtumapohjaisessa tavassa metodia kutsutaan aina, kun kuvakehys on valmis videovirrassa.

On mahdollista myös tallettaa kaikki laitteen antama data tiedostoon. Yleensä tiedostopäätteeksi tulee oni. Tällaista tiedostoa voidaan ohjelmissa käyttää aivan samalla tavalla kuin siinä tapauksessa, että tieto olisi tullut fyysiseltä laitteelta.

API koostuu neljästä pääluokasta, jotka tarjoavat yhteyden laitteeseen, liitännän sensoreihin, työkalut videovirran käsittelyyn ja pääsyn yksittäisiin kehyksiin sekä metadataan. Lisäksi on useita apuluokkia, joiden tarkoituksena on yleensä muuttaa dataa helpommin ymmärrettävään muotoon. [17.]

OpenNI-luokka tarjoaa yhteyden fyysisiin laitteisiin. Sen tärkeimmät metodit ovat intialize() ja shutdown(), Initialize-metodia on kutsuttava OpenNI API:a käyttävässä ohjelmassa aina ensimmäiseksi. Metodi alustaa kaikki laiteajurit ja hakee kaikki saatavilla olevat OpenNI yhteensopivat laitteet. Shutdown-metodia on kutsuttava aina ennen ohjelman lopetusta, jotta kaikki laiteajurit tulevat suljetuksi ja käytetty muisti vapautetaan.

Luokka pitää sisällään perustason yhteyden videovirtaan, tapahtumapohjaisen laitteenkäsittelymahdollisuuden, virheilmoitusten muuntamisen ihmisen ymmärtämään muotoon ja tiedot API:n versiosta. Mahdolliset tapahtumat ovat laitteen kytkeminen, laitteen poisto ja laitteen tilan muutos. [17.]

Device-luokka mahdollistaa liitännän yksittäiseen fyysiseen laitteeseen laiteajurin kautta. Sillä voidaan tehdä liitäntä myös laitetta simuloivaan tiedostoon. Tämän luokan olioita käytetään fyysisen laitteen tai tiedoston yhdistämiseen ja asetusten muokkaukseen. Yhdistämisen jälkeen voidaan muodostaa videovirta kyseisestä laitteesta tai tiedostosta.

Laite avataan open()-metodilla, jolle annetaan parametriksi laitteen URI eli laitteen identifioiva merkkijono (engl. Uniform Resource Identifier). Mikäli voidaan varmuudella tietää, että järjestelmään on kytketty vain yksi laite, URI:na voidaan käyttää merkkijonoa ANY_DEVICE. Mikäli laitetta simuloidaan tiedostolla, URI:ksi annetaan tiedoston polku. Avattu laite on lopuksi suljettava close()-metodilla, jotta laite ja laiteajuri jäisivät oikeaan tilaan niitä käyttävää seuraavaa sovellusta varten.

Edellisten lisäksi luokka sisältää metodeja, joilla voidaan lukea, mitä sensoreita laitteessa on sekä muita laitekohtaisia ja USB-liitännän tietoja. Tähän luokkaan kuuluvat myös metodit eri kameroiden muodostamien kuvien asettamiseksi päällekkäin ja kuvakehysten synkronoimiseksi. [17.]

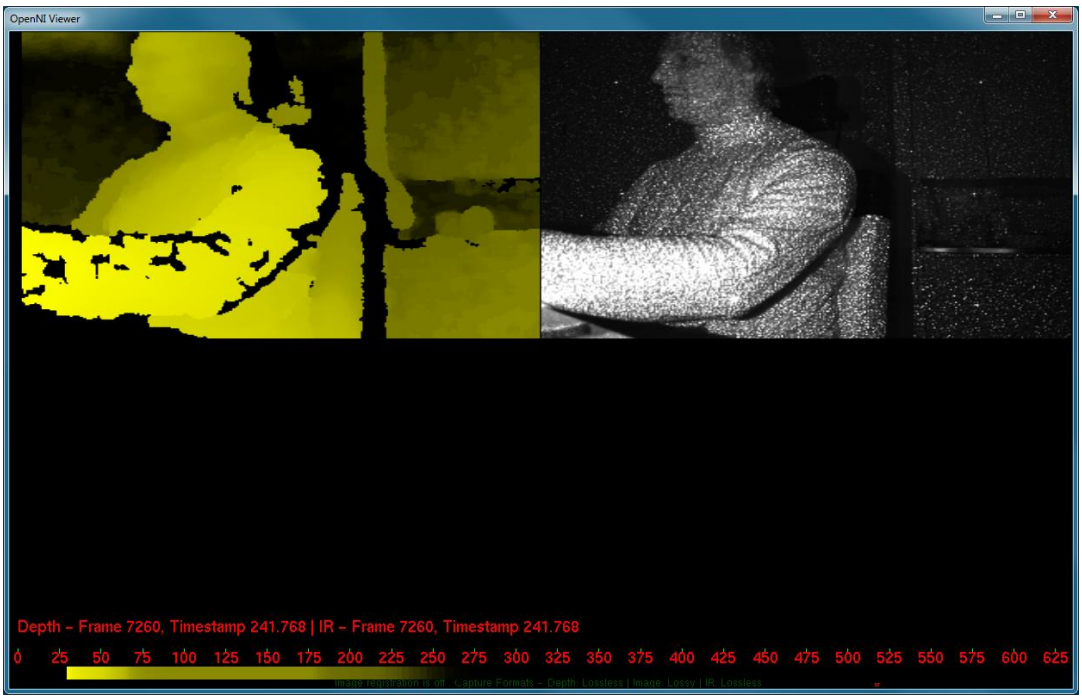
VideoStream-luokka kapseloi Device-luokan muodostaman datavirran. Sen metodien avulla voidaan asettaa videovirran parametrit sekä aloittaa ja lopettaa videovirta.

Luokka sisältää metodit kehysnopeuden, resoluution ja pikseleiden tyypin säätämiseksi. Kuva voidaan kääntää peilikuvaksi tai rajata, mikäli laite tukee näitä ominaisuuksia. Laitteessa voi olla myös muita valmistajakohtaisia asetuksia, joihin päästään käsiksi tämän luokan kautta. [17.]

VideoFrameRef-luokka kapseloi videovirrasta luetun yksittäisen kuvakehyksen. Se mahdollistaa pääsyn taulukkoon, johon kehyksen kuvatieto on tallennettu sekä kehyksessä tarvittavaan muuhun informaatioon, kuten resoluutioon, kuvakokoon ja niin edelleen. Kuvatieto voi tulla RGB- tai syvyyskameralta. [17.]

3.3 Asennus ja käyttö

Ohjelmointikehitysympäristön asennus aloitetaan lataamalla omaa käyttöjärjestelmää vastaava asennuspaketti. Paketti sisältää asennusohjelman, joka asentaa tarvittavat laiteajurit ja ohjelmiston. Ohjelmiston mukana tulee muutamia esimerkkiohjelmia lähdekoodeineen. Esimerkkiohjelmat ovat myös valmiiksi käännettynä suoritettavaan muotoon. Lisäksi SDK:n mukana toimitetaan dokumentaatio HTML-muodossa sekä NiViewer-työkaluohjelma, jolla voidaan testata useimpia OpenNI:n ominaisuuksia. NiViewer-ohjelman toimintaa on esitelty kuvaruutukaappauksella kuvassa 4. Ohjelman oman käyttöohjeen saa esille painamalla kysymysmerkkiä. [13.]



Kuva 4. NiViewerin näkymä, jossa syvyysanturin ja infrapunakameran kuva.

Redist-hakemisto sisältää kaikki tiedostot, jotka tarvitaan jokaisessa OpenNI:tä käyttävässä sovelluksessa. Hakemiston sisältö tulee kopioida jokaisen sovelluksen siihen hakemistoon, jossa sovelluksen käynnistävä ohjelma on. [13.]

Asennus 64-bittisellä *Windows 7-käyttöjärjestelmällä* varustettuun testikoneeseen sujui täysin ilman ongelmia. Myös C++-kielisen esimerkkiohjelman kääntäminen Visual Studio 2012 Professional -ohjelmistolla tapahtui ongelmitta, vaikka ohjeet olivatkin Visual Studio 2010:lle.

Linux-järjestelmään ohjelmisto ja laiteajurit asentuivat myös täysin ongelmitta ja laite toimi valmiiksi käännetyillä esimerkkiohjelmilla moitteetta. Kääntämisohjeisiin kaipaisi kuitenkin edes jonkinlaista esimerkkiä, koska komentorivipohjainen kääntäjä on melko vaikea käyttää. Esimerkkiohjelma jäikin tässä tapauksessa kääntämättä. Testikoneessa oli käyttöjärjestelmänä 64-bittinen Ubuntu 12.04.

Ongelmat

Aikaisemmista versioista poiketen OpenNI 2.0:sta alkaen SDK:ssa ei enää ole tukea C#-ohjelmointikielelle vaan ainoastaan C++:lle. Jos siis halutaan käyttää C#-ohjelmointikieltä, joudutaan käyttämään väliohjelmistoja.

Luokkien ja metodien toiminta on selostettu melko lyhyesti, eikä monien niihin liittyvien vakioiden ja parametrien toiminnasta ole minkäänlaista kuvausta. Esimerkiksi PIXEL_FORMAT_DEPTH_1_MM- ja PIXEL_FORMAT_DEPTH_100_UM-nimisten vakioiden eroa ei kerrota yhtään missään, vaan se jää käyttäjän kokeiltavaksi. [13.]

OpenNI 2.2 Beta -versioon on lisätty Java-kääre (engl. Wrapper), joka mahdollistaa SDK:n käyttämisen Java-ohjelmointikielellä. Minkäänlaisia aloitusohjeita ei Javalle sivustolla kuitenkaan ole. Ohjeet luokkien ja metodien käyttöön tulevat ladattavan ohjelmistopaketin mukana, eli ne on etsittävä SDK:n asennuskansiosta. [13.]

Esimerkkiohjelmissa ei ole juurikaan kommentointia, mikä vaikeuttaa niiden toiminnan ymmärtämistä. Java-kielisiä esimerkkejä ei ole kuin yksi, eikä sekään tunnu toimivan kunnolla vaan jää usein jumiin. Ohjelman jumiutuminen johtuu ilmeisesti siitä, että siinä käytetty kehyksen valmistumisen aiheuttava tapahtumankäsittelijä on eri säikeessä kuin Swing-käyttöliittymäkomponenttien tapahtumankäsittelijä. Tämä rikkoo Swingin yhden säikeen sääntöä ja aiheuttaa ohjelman jumiutumisen silloin, kun jotain asiaa käyttöliittymällä yritetään muuttaa. Java-versiossa myöskään monet parametrit, kuten esimerkiksi edellä mainittu ANY_DEVCE, eivät toimi samalla tavoin kuin C++-ohjelmointikieltä käytettäessä. [18, s. 416.]

Ohjaimessa voi olla mikrofonit, mutta SDK ei sisällä mitään työkaluja, joilla tätä ominaisuutta voisi hyödyntää. [13.]

3.4 OpenNI SDK ja Kinect SDK

Kinect SDK:n avulla saadaan käyttöön ominaisuuksia, kuten luurankomalli, eleiden tunnistus ja äänikomennot, joita varten OpenNI:ssä on käytettävä väliohjelmistoja esimerkiksi PrimeSensen Niteä. [19; 20.]

OpenNI:n tarkoituksena on siis päästä käsiksi suoraan ohjaimen antamaan dataan, mutta Kinect SDK tarjoaa sekä kehittyneemmät työkalut, joiden avulla voidaan tunnistaa erilaisia käsien ja vartalon liikkeitä, että jonkinlaisen suoran pääsyyn kameroiden dataan. [17;19.]

Koska Kinect SDK on Microsoftin kehittämä, sitä voi käyttää vain yhtiön omissa käyttöjärjestelmissä. Tuetut käyttöjärjestelmät ovat Windows 7 ja Windows 8, molemmat ovat tuetut sekä 32- että 64-bittisinä. Avoimeen lähdekoodiin perustuvasta OpenNI:stä sen sijaan on olemassa versiot myös Linuxille ja Applen OS X:lle. Linux-jakeluista virallisesti tuettu on tosin ainoastaan Ubuntu. [15;16.]

4 OpenNI:hin perustuvat sovellukset

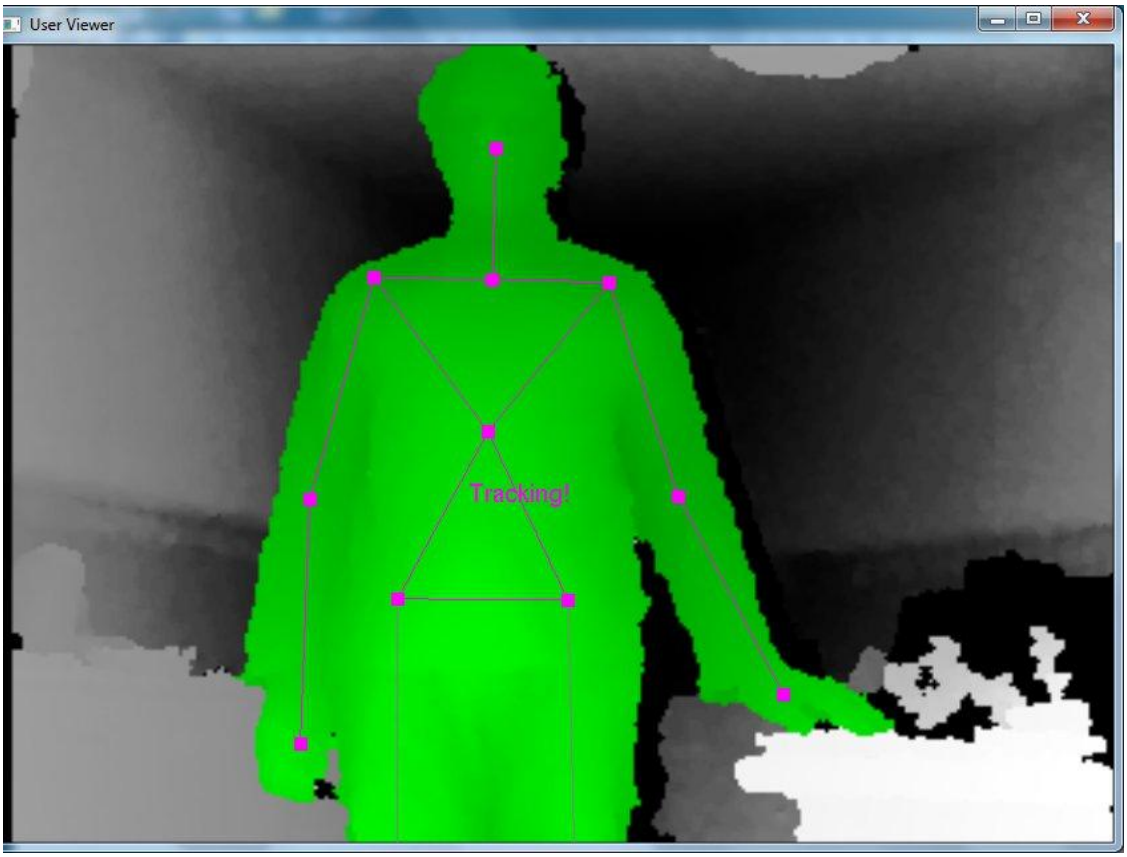
OpenNI:n sivustolta on ladattavissa SDK:n lisäksi monenlaisia väliohjelmistoja, työkaluohjelmistoja ja sovelluksia. Määritelmät eivät kuitenkaan ole tarkkoja vaan monet ohjelmistot on laitettu sivustolla useamman kategorian alle. Itse SDK:n voi ladata vapaasti, mutta kaikkien muiden ohjelmistojen lataaminen vaatii yleensä rekisteröitymisen OpenNI-yhteisön jäseneksi. Osa ohjelmistoista on myös vapaasti ladattavissa GitHub-sivustolta. [21]

4.1 Väliohjelmat

Väliohjelmat ovat ohjelmakirjastoja, jotka toimivat alemman tason kerroksen tai systeemikerroksen ja varsinaisen sovelluksen välittäjinä. Väliohjelmakirjastot helpottavat ohjelmistokehittäjien työtä toteuttamalla alemman tason prosesseja. [22.]

PrimeSense Nite

Suosituin ja ehkä tärkein väliohjelmakirjasto on PrimeSense Ltd:n kehittämä Nite-ohjelmisto. Se tarjoaa ohjelmointirajapinnan kädellä tai kokovartalolla tapahtuvaan liiketunnistusohjaukseen sekä pääsyn laitteen mikrofonien antamaan ääneen. Ohjelmistossa on työkalut käden paikannukseen ja seuraamiseen, käyttäjän erottamiseen taustasta, luurankomallin seuraamiseen, useiden eleiden tunnistukseen ja paljon muuhun. Kuvassa 5 nähdään kuvaruutukaappauksena ohjelmistopaketin mukana tulevan esimerkkiohjelman toimintaa, jossa käyttäjä erotetaan mustavalkoisesta taustasta värillisenä ja hänestä tehdään luurankomalli. [20; 21.]



Kuva 5. Niten luurankomalli.

Nite on tehty alun perin C++-ohjelmointikielelle, mutta 2.2 versiosta alkaen siihen on lisätty tuki myös Javalle. Käyttöjärjestelmäversioita Nite 2.2:een on Windowsille, Linuxille ja Mac OS:lle. Sivustolla on maininta myös tuesta Androidille, mutta latauslinkkiä, josta Android-version voisi ladata, ei kuitenkaan ole. [20.]

Toimiakseen Nite 2.2 tarvitsee OpenNI:stä 2.2 version. Molemmat ohjelmistot ovat Beta-versioita mikä tarkoittaa sitä, että niissä voi vielä olla epävakautta.

Muut väliohjelmakirjastot

Monia muitakin käsien, sormien, vartalon ja niiden luurakomallin tai eleiden tunnistavia ohjelmistoja on olemassa. PrimeSense on kehittänyt myös ohjelmiston, joka tunnistaa tarttumis- ja vapautuseleet. Latausmäärien perusteella nämä muut ohjelmistot eivät ole läheskään niin suosittuja kuin NIte. [20.]

Ohjelmistoja, jotka käyttävät liiketunnistinohjainta 3D-skannerina on olemassa muutama. Lisäksi on olemassa Southern California-yliopiston kehittämä RGB-kmeraa ja syvyysanturia hyödyntävä ohjelmisto, jota voi käyttää käyttäjän tunnistamiseen. [20.]

Lisäksi on olemassa työkaluohjelmistoja, kuten ohjelmisto musiikin tekemiseen, Unity3D-ohjelmistoa tukeva sovellus sekä pari kappaletta sovelluksia, jotka mahdollistavat OpenNI:n käytön C#-ohjelmointikielellä. [21.]

4.2 Sovellukset

Avoimen koodin projekteille tyypillisesti myös OpenNI:lle on kehitetty hyvin monenlaisia sovelluksia. Sovelluksia on liiketunnistinohjaimen käyttämiseksi 3D-skannerina, jolloin ohjaimen antamasta kuvadatasta voidaan muodostaa 3D-mallinnus jotain 3D-mallinnusohjelmaa varten. On myös vartalon liikkeiden ja eleiden seurantaohjelmia, kasvojentunnistusohjelmia ja jopa ohjelma, joka laskee yleisön määrän tunnistamalla kuvasta kasvojen lukumäärän. [21.]

Asus Xtion -ohjaimen mukana toimitetaan kolme peliohjelmaa, ja lisää sovelluksia voi ladata Xtion Store -sivustolta. Sivustolla on sekä maksullisia että ilmaisia sovelluksia. Osan sovelluksista ilmoitetaan toimivan ainoastaan Xtion-ohjaimella. [23; 24.]

5 Asus Xtion

5.1 Esittely

Asus Xtion, jonka alkuperäinen nimi oli Wavi Xtion, on Asuksen ja PrimeSensen yhteistyönä kotitietokoneita varten kehitetty liiketunnistinohjain. Ohjainta myydään sekä Asus Xtion että PrimeSense Sensor tuotemerkeillä. Koska laite perustuu samaan PrimeSensen kehittämään tekniikkaan kuin Microsoftin Kinect, niiden perustekniikka poikkeaa toisistaan vain hyvin vähän. Xtionissa moottoroitu jalusta on korvattu saranoidulla jalustalla ja ohjelmoitava ledi on jätetty pois. Muita eroja on kameroiden resoluutiossa ja toimintaetäisyyksissä, mutta nämä erot ovat melko pieniä. Xtionin äänen suurin näytteenottotaajuus on suurempi kuin Kinectissä. Kinectissä puolestaan on neljä mikrofonia, kun taas Xtionissa niitä mallista riippuen on vain kaksi tai ei ollenkaan. Lisäksi PrimeSense Sensorista on saatavana versio, joka toimii lyhyemmillä etäisyyksillä kuin Xtion ja Kinect. Microsoftin Xbox One -pelikonsolia varten kehittämä Kinect puolestaan on samasta nimestä huolimatta täysin uusi ja kehittyneempi laite. [1; 4; 6; 25; 26.]

Kuva 6. Asus Xtion. [23.]

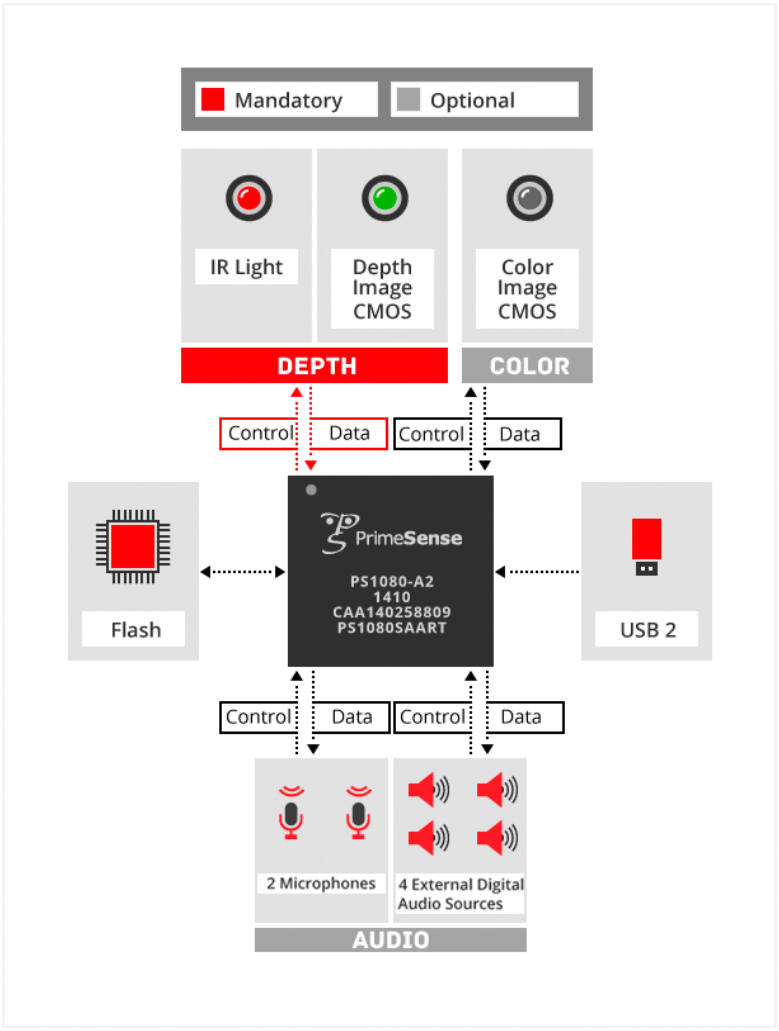
Xtionista on olemassa kolme erilaista versiota: Xtion, Xtion Pro ja Xtion Pro Live. Kaksi ensimmäistä poikkeavat toisistaan vain mukana toimitettavan ohjelmiston osalta. Pro-versio on tarkoitettu kehittäjille, ja sen mukana toimitetaan vain laiteajurit, OpenNI SDK ja Nite -ohjelmistot. Tavallinen versio taas on tarkoitettu käyttäjille, ja sen myyntipakkaus sisältää laiteajureiden lisäksi Asus Xtion Portal -ohjelmiston, jonka avulla ohjainta voi käyttää hiiren tai peliohjaimen korvaajana, sekä käyttöohjekirjat ja kolme peliä. Pro Live-versiossa on RGB-kamera, infrapunakamera, syvyysanturi ja kaksi mikrofonia. Kahdesta muusta versiosta RGB-kamera ja mikrofonit puuttuvat. Erot voi havaita vertailemalla kuvaa 6 ja kuvaa 7, jossa anturin rakenne on selitettynä. [23.]

Kuva 7. Asus Xtion Pro Live. [26.]

5.2 Ominaisuudet

Xtion perustuu PrimeSensen kehittämään teknologiaan ja PS1080-piiriin. Laitteessa on koodattua infrapunavaloa lähettävä valonlähde ja infrapunakamera, jolla kohteesta heijastuva infrapunavalo luetaan. Piirissä itsessään on algoritmit, joiden avulla kohteesta muodostetaan syvyyskuva. Valinnaisena laitteessa voi olla myös tavallista värikuvaa varten RGB-kamera sekä mikrofonit. Toimintakaavio on esitetty kuvassa 8, jossa anturin välttämättömät osat on merkitty punaisella ja valinnaiset harmaalla. Piiri osaa itse synkronoida syvyyskuvan ja värikuvan. Syvyyskuvan muodostamiseen ja kuvien synk-

ronointiin ei siis tarvita lainkaan isäntälaitteen suorittimen laskentatehoa. Laite liitetään tietokoneeseen USB-liitännän kautta. [27.]



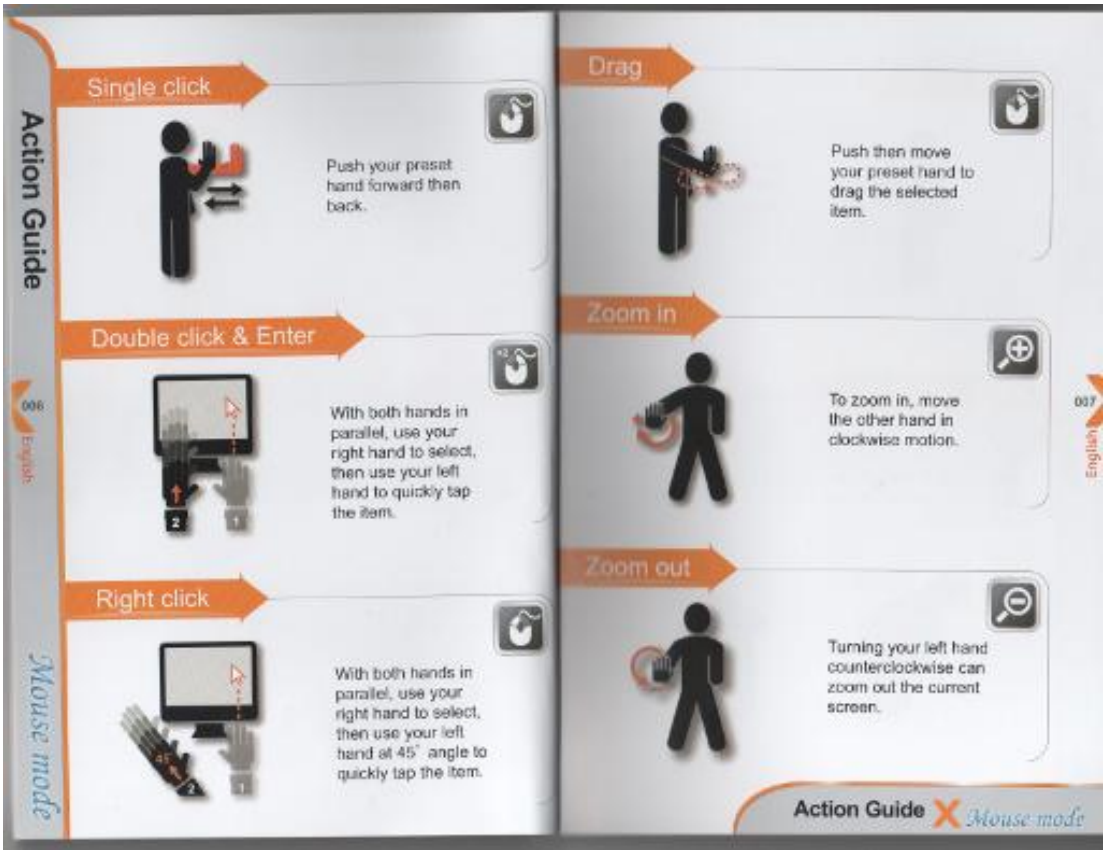
Kuva 8. PS1080:n toimintakaavio. [27.]

5.3 Käyttö

Asus Xtion Portal -ohjelmiston käyttöönotto on melko helppoa. Ohjelma kehottaa aluksi käyttäjää seisomaan kuin kaktus. Tämä tarkoittaa sitä, että käyttäjä seisoo molemmat kädet kohotettuina. Näin ohjelmisto kalibroidaan tunnistamaan käyttäjän ääriviivat. Tämän jälkeen laitetta voi käyttää käyttöjärjestelmän ohjaamiseen. Ohjelmisto myös huomaa, jos laitetta on siirretty ja pyytää käyttäjää tekemään kalibroinnin uudelleen.

Ohjelmistossa on erilaisia toimintatiloja. Esimerkiksi hiiritilassa ohjainta voidaan käyttää hiiren asemasta ja pelitilassa peliohjaimena. Ohjaamiseen tarvittavat erilaiset eleet on

selitetty laitteen myyntipakkaukseen kuuluvassa Action Guide -nimisessä ohjekirjassa. Kuvassa 9 on ohjekirjasta aukeama, jossa kerrotaan, millaiset liikkeet vastaavat hiiren toimintoja. [28.]



Kuva 9. Hiiren toimintoja korvaavia liikkeitä. [28.]

5.4 Ongelmat

Ohjaimen toiminta on melko epämääräistä, joten on vaikeaa osua kuvaruudulla haluttuun kohteeseen, kuten esimerkiksi ikkunan sulkevaan rastiin. Eri liikkeiden tekeminen niin, että ohjelmisto tunnistaisi ne oikein, vaatii melko paljon opettelua. Ohjaaminen onnistuu kunnolla vain seisten, ja välimatkan laitteen ja käyttäjän välillä on oltava noin kaksi metriä, jotta riittävän suuri osa käyttäjästä mahtuisi kameran kuvaan. Tämän vuoksi laitteen käyttämiseen tarvitaan melko paljon tilaa. Kirkas auringonpaiste ja kiiltävät pinnat saattavat myös häiritä laitteen käyttöä.

Koska Asus Xiton portal -ohjelmisto toimii OpenNI:n vanhan version ajureilla, se lakkaa kokonaan toimimasta, kun tietokoneeseen asennetaan OpenNI 2.0 tai uudempi SDK.

6 Esimerkkisovellus

6.1 Esittely

Esimerkkisovelluksen tavoite oli tutkia sitä, kuinka helppoa tai vaikeaa SDK:ta on soveltaa käytännössä sekä sitä, voiko pelkällä OpenNI:llä ilman väliohjelmakirjastoja tehdä yleensäkään mitään. Lisäksi tavoitteena oli selvittää, kuinka paljon ohjaimen epätarkkuudesta on haittaa ja voiko sen vaikutusta jotenkin vähentää. Piirto-ohjelma näytti soveltuvan tähän tarkoitukseen erinomaisesti, koska siinä syntyvän viivan mutkittelusta tai pisteen paikallaan pysymisestä näkee visuaalisesti kuinka paljon epätarkkuutta esiintyy.

Ohjelman ideana on yksinkertainen piirto-ohjelma, jossa kuvan voi piirtää pelkästään huitomalla kädellä ilmaan. Käyttäjä siis osoittaa liiketunnistinta kädellä ja käyttää kättään virtuaalisena siveltimenä. Siveltimen paikka lasketaan ohjaimen antamasta datasta lähimmän pisteen perusteella. Ohjelmassa valitaan ensin nollataso eli se etäisyys, jossa kuviteltu kynä koskettaa kuviteltua piirtopaperia. Kättä nollatasosta lähemmäksi siirtämällä viivan paksuus levenee ja kauemmaksi siirtämällä piirtoa ei tapahdu. Ohjelmassa on mahdollisuus käyttää piirtämiseen muutamaa eri väriä. Piirretty kuva voidaan myös tallentaa kovalevylle tai ladata sieltä aikaisemmin piirretty kuva.

Ohjelmaan on määritelty piirtoalue ja valinta-alueet. Valinta-alueella on yläreunassa keskellä nollatason asetus. Nollatason asetuksen molemmilla puolilla on värin valinta. Käyttäjä valitsee värin ja nollatason osoittamalla vastaavaa neliötä. Vasemmassa reunassa on punainen suorakaide, jota osoittamalla piirto loppuu. Piirtämisen lopetustoiminto on välttämätön, jottei mitään piirrettäisi tarkoituksettomasti, mikäli käyttäjä joutuu siirtymään lähemmäksi ohjainta. Oikeassa alareunassa näytetään neliö, joka näyttää valitun värin. Ohjelma näyttää toisessa ikkunassa kameran kuvaa käyttäjästä ja toisessa piirrettävää kuvaa. Käyttäjän kuvaa näyttävässä kuvassa näytetään pienellä punaisella neliöllä lähin piste eli siveltimen paikka. Näin käyttäjä näkee heti, jos lähimpänä pisteenä on esimerkiksi jonkin huonekalun nurkka. Piirrosta näyttävän ikkunan oikeassa reunassa on lisäksi palkki, joka näyttää syvyysetäisyyden eli etäisyyden asetetusta nollatasosta. Esimerkkisovelluksen lähdekoodi on liitteessä 1.

6.2 Ohjelmointikielen valinta

Esimerkkisovelluksen ohjelmointikieleksi valittiin Java. Java oli tekijälle tutuin, ja sillä on aloittelijankin melko helppoa saada aikaiseksi ihan kelvollinen graafinen käyttöliittymä.

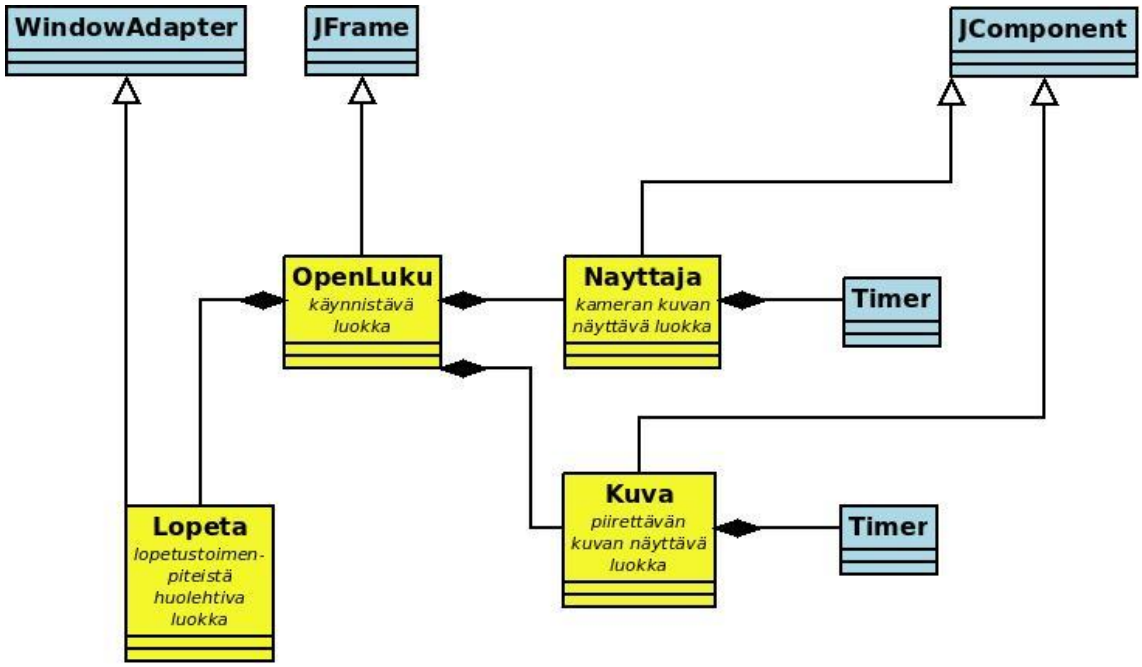
Toinen mahdollisuus olisi ollut C++. C++ on kuitenkin melko haastava ja virhealtis ohjelmointikieli. Graafisen käyttöliittymän Visual Studion, C++:n ja OpenNI:n yhdistelmä osoittautui myös ongelmalliseksi. Graafisen käyttöliittymän käyttäminen aiheutti esimerkiksi OpenNI:n otsikkotiedostoista virheilmoituksia jopa valmiiden esimerkkien lähdekoodia käytettäessä.

Käyttöliittymän olisi voinut tehdä käyttäen OpenGL:ää, jota on käytetty joissakin OpenNI SDK:n mukana tulleissa esimerkeissä. Tämä olisi kuitenkin ollut melko työlästä, koska ensin olisi joutunut opiskelemaan OpenGL:n ja sitten ohjelmoimaan melko paljon itse sellaisia asioita, jotka esimerkiksi Javassa tulevat valmiina.

Käyttöliittymä olisi luultavasti syntynyt C#-ohjelmointikielellä sunnilleen yhtä helposti kuin Javallakin. C#:lle ei kuitenkaan ole OpenNI 2.0:ssa tukea, joten siinä olisi joutunut käyttämään jonkun muun tekemää ohjelmistoa apuna.

6.3 Ohjelman rakenne

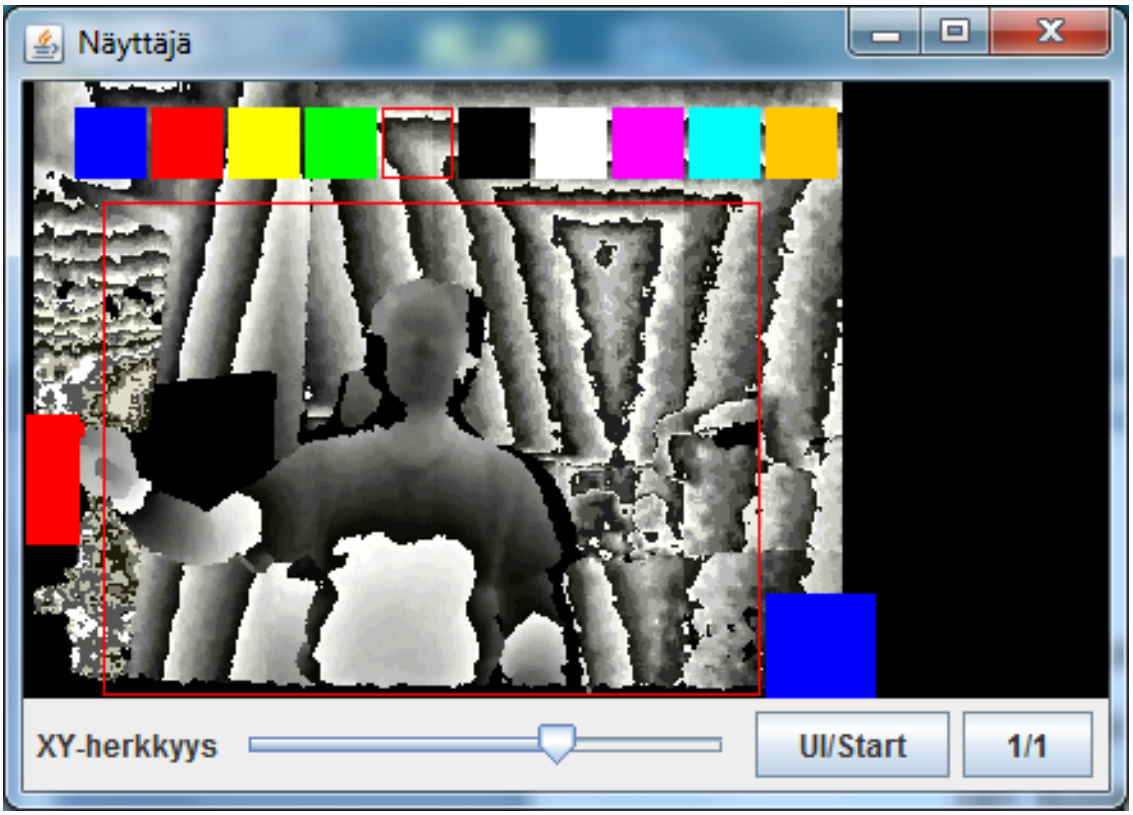
Ohjelmassa on neljä pääluokkaa: OpenLuku, Nayttaja, Kuva ja Lopeta. Ohjelman karkea rakenne on esitetty kuvassa 10 kaaviona. Kuvassa Javan valmiit luokat on väritetty sinisellä ja omat luokat keltaisella. Ohjelman käynnistävä luokka on OpenLuku, ja se on periytetty luokasta JFrame, jotta siitä saadaan graafinen käyttöliittymä. Kuva- ja Nayttaja-luokat on periytetty luokasta JComponent, jotta ne voidaan helposti liittää Javan Swing-käyttöliittymään. Lopeta-luokka on periytetty luokasta WindowAdapter.



Kuva 10. Ohjelman pääluokat kaaviona.

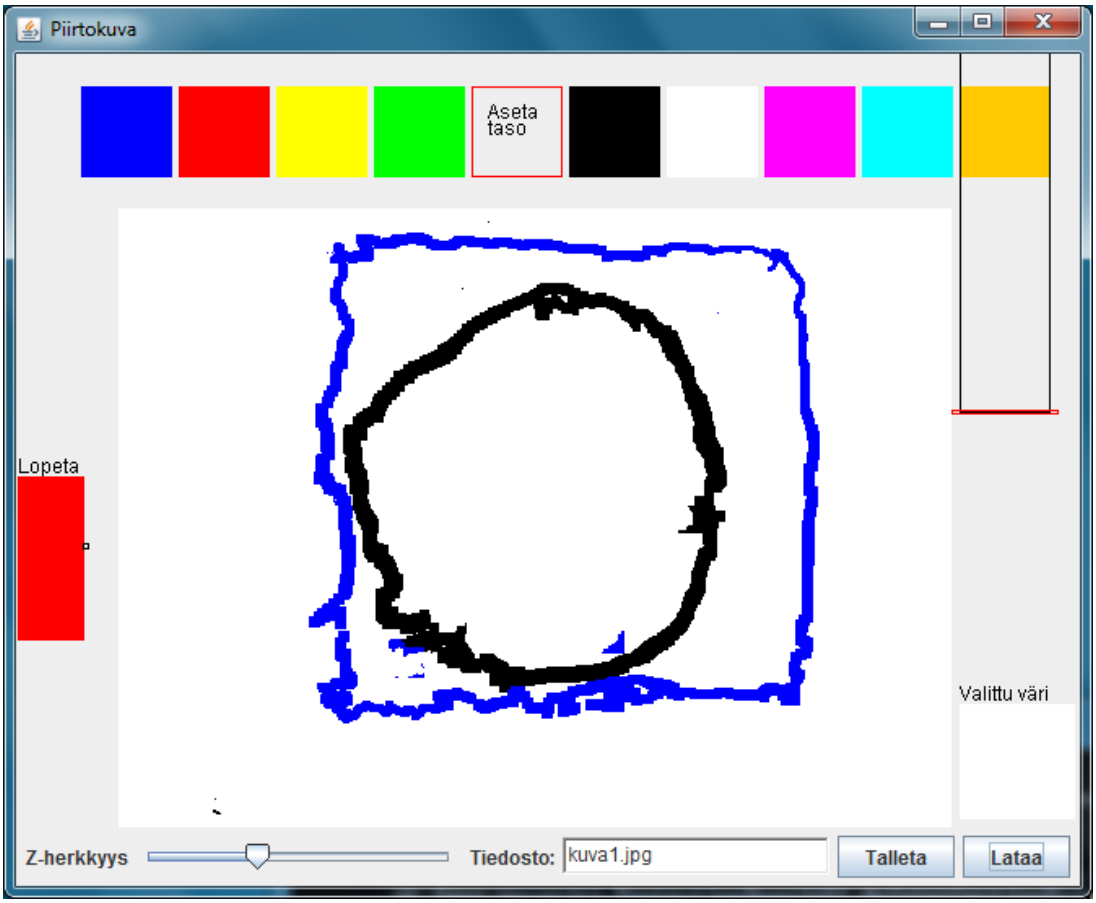
OpenLuku-luokka on käynnistävä luokka, ja sen tehtävänä on näyttää kaksi ikkunaa ja niihin liitetyt käyttöliittymäkomponentit sekä huolehtia käyttöliittymän tapahtumista. Nayttaja- Kuva- ja Lopeta-luokan oliot luodaan tämän luokan rakentajassa. Luokassa on myös metodit piirretyn kuvan tallentamista ja lataamista varten. Luokan Main-metodissa alustetaan OpenNI ja annetaan alustuksen tuloksen perusteella virheilmoitus, mikäli järjestelmään ei ole kytketty yhteensopivaa laitetta. Lopuksi luokasta itsestään muodostetaan olio ja asetetaan se näkyväksi, jolloin Java Swing muodostaa näytölle ikkunat.

Nayttaja-luokan tehtävänä on lukea dataa syvyyskameralta ja muodostaa siitä näytöllä näytettävä kuva sekä laskea piirrettävän pisteen koko ja koordinaatit. Kameran kuvaa näyttävä ikkuna on esitetty kuvassa 11. Luokassa tehdään liiketunnistimen alustamiseen ja sulkemiseen tarvittavat toimenpiteet. Tähän luokkaan on sijoitettu myös piirtämisen aputoiminnot eli värin valinta, nollatason asetus ja piirtämisen lopetus. Piirrettävä piste määritellään laskemalla syvyysanturin datasta lähin piste. Koska lähin piste ei ole tarkka, vaan värisee hieman, muutaman kuvakehyksen lähimmistä pisteistä lasketaan keskiarvo värinän vähentämiseksi. Luokkaan on toteutettu Timer-luokan ajastin, joka aiheuttaa neljänkymmenen millisekunnin välein kuvakehyksen lukemisen ja ikkunan päivittämisen suorittavan tapahtuman.



Kuva 11. Kameran kuvaa näyttävä ikkuna.

Kuva-luokan tehtävänä on muodostaa ja näyttää piirrettävä kuva. Kuva-luokan rakentaja saa parametrina Nayttaja-luokan olion, jotta siitä päästään käsiksi tarvittaviin muuttujiin, kuten lähimmän pisteen koordinaatteihin ja valittuun väriin. Piirrettävää kuvaa näyttävä ikkuna on esitetty kuvassa 12. Ikkunassa näytetään käytön helpottamiseksi samat aputoimintojen kuviot, jotka kameran kuvan näyttävässä ikkunassa ovat sekä syvyyssuuntaisen etäisyyden näyttävä palkki. Myös tässä luokassa on Timer-luokan ajastin, joka aiheuttaa ikkunan päivittämisen suorittavan tapahtuman neljänkymmenen millisekunnin välein.



Kuva 12. Piirrettävää kuvaa näyttävä ikkuna.

Lopeta-luokka on apuluokka, joka suorittaa lopetustoimenpiteet, kun ohjelman pääikkuna eli kameran kuvaa näyttävä ikkuna suljetaan. Lopeta-luokan rakentaja saa parametrina Nayttaja-luokan olion, jotta siitä voidaan kutsua laitteen sulkemistoimenpiteet suorittavaa metodia.

6.4 Ohjelman käyttö

Ohjelman käyttö aloitetaan osoittamalla ylälaidassa keskellä näkyvää punaista kehystä, jolloin ohjelma asettaa sen syvyystason, jossa sivellin koskettaa kangasta. Edellä mainittua pistettä lähemmäksi siirtymällä pisteen koko kasvaa ja kauemmaksi siirtymällä piirtoa ei tapahdu. Pisteen etäisyys asetustasosta näytetään Piirtokuva-ikkunan oikeassa laidassa olevalla palkilla. Piirrettävän pisteen paikka näytetään Näyttäjä-ikkunassa pienellä punaisella neliöllä ja Piirtokuva-ikkunassa pienellä mustalla neliöllä silloin kun sivellin ei kosketa kangasta. Väri valitaan osoittamalla halutun väristä neliötä.

Kuvan piirtäminen lopetetaan osoittamalla vasemmassa laidassa olevaa punaista suorakaidetta, jolloin ajastimet pysähtyvät ja kuvien päivittyminen loppuu. Ajastimet käynnistyvät uudelleen painamalla UI/Start-nappulaa. Edellisen nappulan painaminen muuttaa myös anturin syvyysasetusta, jolloin kameran kuva muuttuu sinisävyiseksi ja anturin syvyysalue laajenee. Asetus palautuu takaisin painamalla samaa nappulaa uudelleen.

Ikkunoissa on liukusäätimet, joilla voidaan säätää herkkyyttä. XY-herkkyyssäädöllä säädetään sitä, kuinka monen kuvakehyksen lähimmästä pisteestä lasketaan keskiarvo. Z-herkkyyssäädöllä säädetään sitä, kuinka suuri syvyyssuuntainen liike vastaa piirrettävän pisteen koon muuttumista.

Kameran kuva näytetään oletuksena neljäsosan kokoisena ja se voidaan muuttaa täysikokoiseksi painamalla 1/1-nappulaa. Kuvan voi muuttaa takaisin pienemmäksi painamalla 1/4-nappulaa.

Piirretyn kuvan voi tallettaa kirjoittamalla tiedoston nimen Tiedosto-kenttään ja painamalla Talleta-nappulaa. Vastaavasti aikaisemmin piirretyn kuvan voi ladata jatkokäsittelyä varten painamalla Lataa-nappulaa.

6.5 OpenNI ohjelmointityössä

OpenNI on hyvin matalan tason järjestelmä, mikä tarkoittaa sitä, että ohjelmoijalle annetaan vain työkalut laitteen hallintaan ja pääsy laitteen antamaan tietovirtaan. Kaikki korkeamman tason asiat, kuten hahmojen tai liikkeiden tunnistus, jäävät joko ohjelmoitavaksi itse tai on käytettävä apuna jonkun muun tekemiä valmiita väliohjelmistoja.

Aloitusopas on tehty ainoastaan Visual Studiolle ja gcc-kääntäjälle, joissa käytetään C++ -ohjelmointikieltä. Valmiit esimerkkiohjelmat toimivat hyvin komentoriviltä, mutta sellaisia esimerkkejä, joissa olisi käytetty Windowsin graafisia ikkunoita, ei ole lainkaan.

Java-ohjelmaan on lisättävä alkuun komento, joka lataa valmiiksi käännetyn OpenNI2.dll-tiedoston, sekä asetus *-Djava.library.path="C:\Program Files\OpenNI2\Redist"* virtuaalikoneen optioksi. Näistä johtuen sovellus ei toimi ilman muutoksia muussa kuin siinä järjestelmässä, jolle kyseinen dll-tiedosto on tehty ja OpenNI:n hakemistopolut asetettu oikein. Esimerkiksi tässä työssä tehty esimerkkisovellus toimii vain 64-

bittisessä Windows-järjestelmässä, jossa OpenNI SDK on asennettu oletushakemistoon. Kyseisiä välttämättömiä asioita ei kerrota missään OpenNI:n ohjeissa, eikä niitä voi päätellä esimerkkiohjelmasta.

7 Yhteenveto

Alun perin pelikonsoleita varten ja peliohjaimeksi kehitetty liiketunnistin osoittautui melko huonosti muuhun käyttöön sopivaksi. Ongelmallista on sen epätarkkuus ja se, että ainakin vanhemmissa malleissa lähin toimintaetäisyys on melko suuri. Muussa kuin peliohjainkäytössä se soveltuukin parhaiten kuvien selaamiseen silloin, kun käytetään isoa näyttöä, esimerkiksi taulutelevisiota, tai muuhun vastaavaan, missä ei ole suurta tarvetta käyttää näppäimistöä samaan aikaan. Laite soveltuukin lähinnä asiasta kiinnostuneille harrastajille. Hiiren tai kosketusnäytön käyttäminen käyttöjärjestelmän ohjaamiseen on kuitenkin niin paljon helpompaa, että niiden korvaajaksi liiketunnistimesta ei ole.

Ohjelmointiympäristönä OpenNI antaa hyvät perustyökalut päästä käsiksi laitteen antamaan raakaan dataan mutta ei mitään apuvälineitä kyseisen datan hyödyntämiseen.

Aloittelevalle ohjelmoijalle oli melko haasteellista päästä ohjelmointityössä edes alkuun, mutta esimerkit lähdekoodeineen auttoivat paljon. Esimerkkien lähdekoodeihin olisi tosin kaivannut enemmän kommentteja helpottamaan niiden ymmärtämistä. Dokumentoinnissa on selitetty luokkien ja metodien toiminta pääpiirteittäin, mutta esimerkiksi joidenkin parametrien käyttö ja vaikutus selviää vasta kokeilemalla.

Tehdyllä esimerkkisovelluksella voi kyllä piirtää, mutta minkään käyttökelpoisen kuvan piirtäminen sillä on melko mahdotonta. Kuten kuvasta 10 voi nähdä, jo yksinkertaisten muotojen, kuten suorakaiteen tai ympyrän piirtäminen, on melko hankalaa. Viivoista tulee aina kiemurtelevia. Lisäksi kuvan laitoja kohti mentäessä lähimmän pisteen paikan epämääräisyys vain näytti lisääntyvän. Mikäli siis halutaan tehdä vähänkin monimutkaisempia kyseistä ohjainta käyttäviä sovelluksia, kannattaa käyttää jonkun muun tekemiä väliohjelmistoja, jotka tarjoavat korkeamman tason työkalut.

Työ oli kuitenkin opettavainen. Asian luonteesta ja uutuudesta johtuen siinä joutui etsimään tietoa monista eri lähteistä lähinnä ulkomaisilta verkkosivuilta ja miettimään niiden käyttökelpoisuutta. Sovelluksen saaminen toimivaksi edellytti myös monien asioiden hakemista ja löytämistä eri verkkosivuilta ja jopa keskustelupalstojen pirstaleisen

tiedon yhdistelemistä. Oppimismielessä työn parasta antia oli ehkä kuitenkin käyttöliittymän ja liikkuvan grafiikan tekeminen Javalla. Työ tehtiin omatoimisesti vapaa-ajalla, joten motivaation ja ajan löytäminen sen tekemiseen oli välillä haastavaa.

Lähteet

1 Kinect . 2013 Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Kinect>. Luettu 18.3.2013.

2 Kinect for Windows SDK to Arrive Spring 2011. 2011. Verkkodokumentti. Microsoft. <http://blogs.technet.com/b/microsoft_blog/archive/2011/02/21/kinect-for-windows-sdk-to-arrive-spring-2011.aspx> Luettu 18.3.2013.

3 Kinetct hakkeroitiin irti Xboxista. 2010. Verkkodokumentti. Digitoday. <http://www.digitoday.fi/data/2010/11/12/kinect-hakkeroitiin-irti-xboxista/201015812/66> Luettu 18.3.2013.

4 OpenNI 3D sensors 2013. Verkkodokumentti. OpenNI. <http://www.openni.org/3d-sensors> Luettu 18.3.2013.

5 About openni. 2013. Verkkodokumentti. OpenNI. <http://www.openni.org/about>. Luettu 18.3.2013.

6 OpenNI. 2013. Verkkodokumentti. OpenNI. <http://en.wikipedia.org/wiki/OpenNI>. Luettu 18.3.2013.

7 Motion controller. 2013. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Motion_controller> Luettu 18.3.2013.

8 Natural user interface. 2013. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Natural_user_interface> Luettu 19.3.2013.

9 Wii Remote 2013. Verkkodokumentti. Wikipedia <http://en.wikipedia.org/wiki/Wii_Remote> Luettu 19.3.2013.

10 PlayStation Eye. 2013. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/PlayStation_Eye> Luettu 19.3.2013.

11 EyeToy. 2013. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/EyeToy> Luettu 19.3.2013.

12 PlatStation Move. 2013. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/PlayStation_Move> Luettu 19.3.2013.

13 . Reference Guide. 2013. Verkkodokumentti. OpenNI <http://www.openni.org/reference-guide> Luettu 20.3.2013.

14 Apache License, Version 2.0. 2013. Verkkodokumentti. The Apache Software Foundation. <http://www.apache.org/licenses/LICENSE-2.0> Luettu 20.3.2013.

15 OpenNI SDK release notes. 2013. Verkkodokumentti. OpenNI <http://www.openni.org/openni-sdk/openni-sdk-release-notes> Luettu 16.11.2013.

16 OpenNI SDK 2013. Verkkodokumentti. OpenNI. <http://www.openni.org/openni-sdk> Luettu 20.3.2013.

17 OpenNI programmer's guide. 2013. Verkkodokumentti. OpenNI. <http://www.openni.org/openni-programmers-guide> Luettu 21.3.2013.

18 Westerholm Mika & Kyppö Jorma. 2004. Java-Ohjelmointi. Jyväskylä: Talentum Media Oy

19 Kinect for Windows features. 2013 Verkkodokumentti. Microsoft. <http://www.microsoft.com/en-us/kinectforwindows/Discover/Features.aspx> Luettu 25.3.2013.

20 NiTE 2.2.0.11. 2014. Verkkodokumentti. OpenNI. <http://www.openni.org/files/nite> Luettu 25.1.2014.

21 Arena. 2013. Verkkodokumentti. OpeNI. <http://www.openni.org/software> Luettu 26.3.2013.

22 Middleware library guidelines. 2013. Verkkodokumentti. OpeNI. <http://www.openni.org/middleware-library-guidelines> Luettu 26.3.2013.

23 Xtion. 2013. Verkkodokumentti. Asus. <http://www.asus.com/Multimedia/Xtion> Luettu 28.3.2013.

24 Xtion Store. 2013. Verkkodokumentti. Asus. <http://www.xtionstore.com> Luettu 28.3.2013.

25 Kinect for Windows Sensor Components and Specifications. 2014. Verkkodokumentti. Microsoft. <http://msdn.microsoft.com/en-us/library/jj131033.aspx> Luettu 25.1.2014.

26 Xtion PRO LIVE. 2014. Verkkodokumentti. Asus. <http://www.asus.com/Multimedia/Xtion_PRO_LIVE> Luettu 25.1.2014.

27 Technology. 2013. Verkkodokumentti. PrimeSense. <http://www.primesense.com/solutions/technology> Luettu 28.3.2013

28 Asus Xtion laitteen mukana toimitetut ohjekirjat.

Sovelluksen lähdekoodi

OpenLuku-luokka

```
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JSlider;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import org.openni.*;

public class OpenLuku extends JFrame{

    private Nayttaja nay;
    private int korkeus=550,leveys=640;
    private JFrame kehys;
    private JPanel pan,pan2;
    private Lopeta loppu;
    private JButton butt,butt2,butt3,butt4;
    private JSlider liuku,liuku2;
    private TextField tiedosto;
    private JLabel nimi,nimi2,nimi3;
    private Kuva kuv;

    private OpenLuku() {
        kehys = new JFrame();
        pan = new JPanel();
        pan2 = new JPanel();
        butt = new JButton();
        butt2 = new JButton();
        butt3 = new JButton();
        butt4 = new JButton();
        liuku = new JSlider(1, 15);
        liuku2 = new JSlider(1, 40);
```

```
nimi=new JLabel("XY-herkkyys");
nimi2=new JLabel("Z-herkkyys");
nimi3=new JLabel("Tiedosto:");
tiedosto = new TextField("kuva1.jpg",20);
liuku2.setToolTipText("Syvyysherkkyys");
butt.setText("Talleta");
butt2.setText("UI/Start");
butt3.setText("1/1");
butt4.setText("Lataa");
nay = new Nayttaja();
kuv = new Kuva(nay);
loppu = new Lopeta(nay);
nay.setStream();
this.addWindowListener(loppu);
this.setSize(120+leveys/2,40+korkeus/2);
this.setTitle("Näyttäjä");
kehys.setTitle("Piirtokuva");

liuku.setValue(10);
liuku2.setValue(15);
pan.add(nimi);
pan.add(liuku);

pan.add(butt2);
pan.add(butt3);
pan2.add(nimi2);
pan2.add(liuku2);
pan2.add(nimi3);
pan2.add(tiedosto);
pan2.add(butt);
pan2.add(butt4);
kehys.setSize(670,550);
kehys.setLocation(leveys/2+121,0);
this.add("Center", nay);
kehys.add("Center",kuv);
this.add("South",pan);
kehys.add("South",pan2);
kehys.setVisible(true);

butt.addActionListener(new ActionListener(){

   @Override
   public void actionPerformed(ActionEvent ae) {
     talletaKuva();
   }

});
butt2.addActionListener(new ActionListener(){
```

```
        @Override
        public void actionPerformed(ActionEvent ae) {
          aloitaUusi();
        }

    });
    butt3.addActionListener(new ActionListener(){

        @Override
        public void actionPerformed(ActionEvent ae) {
          vaihdaKoko();
        }

    });
    butt4.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent ae) {
           lataaKuva();
        }
    });
    liuku.addChangeListener(new ChangeListener() {

        @Override
        public void stateChanged(ChangeEvent ce) {
          nay.herkkyys=liuku.getValue();
           System.out.println(nay.herkkyys);
        }
    });
    liuku2.addChangeListener(new ChangeListener() {

        @Override
        public void stateChanged(ChangeEvent ce) {
           int val=liuku2.getValue();
           kuv.herkkyys=val;
           nay.herkkyysz=val;


        }
    });

}

private void talletaKuva(){
    nay.ajastin.stop();
    BufferedImage kuva = kuv.getBKuva();
    try {
        ImageIO.write(kuva, "jpg",new File(tiedosto.getText()));
    } catch (IOException ex) {
```

```
            Logger.getLogger(OpenLuku.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(null, "Ei voi kirjoittaa tiedostoa: "+tiedosto.getText(), "Virhe", JOptionPane.ERROR_MESSAGE);
        }
        kuv.setBKuva(null);
    }
    private void lataaKuva(){
        nay.ajastin.stop();
        nay.asetus=0;
        kuv.setBKuva(null);
        BufferedImage kuva;
        try {
            kuva=ImageIO.read(new File(tiedosto.getText()));

        } catch (IOException ex) {
            Logger.getLogger(OpenLuku.class.getName()).log(Level.SEVERE, null, ex);
            kuva=null;
            JOptionPane.showMessageDialog(null, "Ei löydy kuvatiedostoa: "+tiedosto.getText(), "Virhe", JOptionPane.ERROR_MESSAGE);
        }
        kuv.setBKuva(kuva);
        nay.ajastin.start();
    }
    private void aloitaUusi(){
        kehys.setVisible(true);
        nay.asetus=0;
        nay.ajastin.start();
        if(nay.ui1){
            nay.mode.setPixelFormat(PixelFormat.DEPTH_100_UM);
            nay.virta.setVideoMode(nay.mode);
            nay.ui1=false;
        }else{
            nay.mode.setPixelFormat(PixelFormat.DEPTH_1_MM);
            nay.virta.setVideoMode(nay.mode);
            nay.ui1=true;
        }
    }
    private void vaihdaKoko(){
        if(nay.puoli==1){
          nay.puoli=2;
          butt3.setText("1/1");
          nay.teePuoli();
          this.setSize(120+leveys/2,40+korkeus/2);
          nay.bKuva=null;
        }else{
            nay.puoli=1;
            butt3.setText("1/4");
            nay.teePuoli();
            this.setSize(leveys, korkeus);
```

```
            nay.bKuva=null;
        }
    }

    public static void main(String[] args) {

        try {
            System.load("C:/Program Files/OpenNI2/Redist/OpenNI2.dll");
        }
        catch (Exception e)
        {
            System.out.println("OpenNI2.dll ei löydy");
            JOptionPane.showMessageDialog(null, "OpenNI2.dll ei löydy", "Virhe", JOptionPane.ERROR_MESSAGE);
        }
      OpenNI.initialize();
        List<DeviceInfo> devicesInfo = OpenNI.enumerateDevices();
        if (devicesInfo.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Laitetta ei ole kytketty!", "Virhe", JOptionPane.ERROR_MESSAGE);
            return;
        }
        final OpenLuku ol = new OpenLuku();
        ol.setVisible(true);
    }
}
```

Nayttaja-luokka

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.*;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.util.ArrayList;
import javax.swing.JComponent;
import javax.swing.Timer;
import java.util.List;

import org.openni.*;

public class Nayttaja extends JComponent {

    private Device xtion;
    protected VideoStream virta;
    private VideoFrameRef raami;
    protected VideoMode mode;
    private int[] kuva, piirtokuva;
```

```
    private int[][] kuva2;
    private int leveys, korkeus, leveys2, korkeus2;
    protected int lahinx, lahiny, lahinz;
    protected BufferedImage bKuva;
    private ArrayList<SensorType> laiteSensorit;
    private PixelFormat pxf = PixelFormat.DEPTH_1_MM;
    protected int asetus;
    protected Color vari = new Color(0xff, 0xff, 0xff, 0xff);
    protected int herkkyys = 10, herkkyysz = 10;
    private int[][] lahimmat;
    private int merkki = 0;
    protected int puoli = 2;
    private int kleveys = 640, kkorkeus = 480, fps = 30, palkkix = 40 / puoli, palkkiy = 20 / puoli, palkkik = 55 / puoli, palk-
kiv = 60 / puoli;
    protected boolean ui1 = true;
    private int viive = 40;
    Timer ajastin = new Timer(viive, new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {
            paivitaKuva(ae);
        }
    });

    public Nayttaja() {

        List<DeviceInfo> devicesInfo = OpenNI.enumerateDevices();
        xtion = Device.open(devicesInfo.get(0).getUri());
        laiteSensorit = new ArrayList<>();

        if (xtion.getSensorInfo(SensorType.DEPTH) != null) {
            laiteSensorit.add(SensorType.DEPTH);
        }
        SensorType type = laiteSensorit.get(0);

        virta = VideoStream.create(xtion, type);
        java.util.List<VideoMode> supportedModes = virta.getSensorInfo().getSupportedVideoModes();
        mode = supportedModes.get(0);
        mode.setResolution(kleveys, kkorkeus);
        mode.setPixelFormat(pxf);
        mode.setFps(fps);
        virta.setVideoMode(mode);

        ajastin.start();

    }

    public synchronized void paintComponent(Graphics g) {

        if (raami == null) {
```

```
    return;
}
g.drawImage(bKuva, 0, 0, this);
Graphics2D g2 = (Graphics2D) g.create();

Rectangle alue = new Rectangle(512 / puoli, 384 / puoli);
alue.setLocation(63 / puoli, 95 / puoli);
g2.setColor(Color.red);
g2.draw(alue);

Rectangle sininen = new Rectangle(palkkik, palkkik);
sininen.setLocation(palkkix, palkkiy);
g2.setColor(Color.blue);
g2.fill(sininen);
g2.draw(sininen);

Rectangle punainen = new Rectangle(palkkik, palkkik);
punainen.setLocation(palkkix + palkkiv, palkkiy);
g2.setColor(Color.red);
g2.fill(punainen);
g2.draw(punainen);

Rectangle keltainen = new Rectangle(palkkik, palkkik);
keltainen.setLocation(palkkix + palkkiv * 2, palkkiy);
g2.setColor(Color.yellow);
g2.fill(keltainen);
g2.draw(keltainen);

Rectangle vihrea = new Rectangle(palkkik, palkkik);
vihrea.setLocation(palkkix + palkkiv * 3, palkkiy);
g2.setColor(Color.green);
g2.fill(vihrea);
g2.draw(vihrea);

Rectangle musta = new Rectangle(palkkik, palkkik);
musta.setLocation(palkkix + palkkiv * 5, palkkiy);
g2.setColor(Color.black);
g2.fill(musta);
g2.draw(musta);

Rectangle valkoinen = new Rectangle(palkkik, palkkik);
valkoinen.setLocation(palkkix + palkkiv * 6, palkkiy);
g2.setColor(Color.white);
g2.fill(valkoinen);
g2.draw(valkoinen);

Rectangle magenta = new Rectangle(palkkik, palkkik);
magenta.setLocation(palkkix + palkkiv * 7, palkkiy);
g2.setColor(Color.magenta);
```

```
g2.fill(magenta);
g2.draw(magenta);

Rectangle syaani = new Rectangle(palkkik, palkkik);
syaani.setLocation(palkkix + palkkiv * 8, palkkiy);
g2.setColor(Color.cyan);
g2.fill(syaani);
g2.draw(syaani);

Rectangle oranssi = new Rectangle(palkkik, palkkik);
oranssi.setLocation(palkkix + palkkiv * 9, palkkiy);
g2.setColor(Color.orange);
g2.fill(oranssi);
g2.draw(oranssi);

Rectangle lopeta = new Rectangle(40 / puoli, 100 / puoli);
lopeta.setLocation(1, palkkiy * 13);
g2.setColor(Color.red);
g2.fill(lopeta);
g2.draw(lopeta);

Rectangle kohta = new Rectangle(5, 5);
kohta.setLocation(lahinx / puoli - 2, lahiny / puoli - 2);
g2.setColor(Color.red);
g2.fill(kohta);
g2.draw(kohta);

Rectangle aseta = new Rectangle(palkkik, palkkik);
aseta.setLocation(palkkix + palkkiv * 4, palkkiy);
g2.setColor(Color.red);
g2.draw(aseta);

Rectangle valittu = new Rectangle(palkkik + 15, palkkik + 15);
valittu.setLocation(palkkix + palkkiv * 9, palkkiy * 20);
g2.setColor(vari);
g2.fill(valittu);
g2.draw(valittu);

if (kohta.intersects(aseta)) {
   asetus = lahinz;
}

if (kohta.intersects(sininen)) {
   vari = Color.blue;
}
if (kohta.intersects(punainen)) {
   vari = Color.red;

}
```

```
    if (kohta.intersects(keltainen)) {
        vari = Color.yellow;
    }
    if (kohta.intersects(vihrea)) {
        vari = Color.green;
    }
    if (kohta.intersects(musta)) {
        vari = Color.black;
    }
    if (kohta.intersects(valkoinen)) {
        vari = Color.white;
    }
    if (kohta.intersects(magenta)) {
        vari = Color.magenta;
    }
    if (kohta.intersects(syaani)) {
        vari = Color.cyan;
    }
    if (kohta.intersects(oranssi)) {
        vari = Color.orange;
    }
    if (kohta.intersects(lopeta)) {
        ajastin.stop();
        asetus = 0;
    }
}

protected void setStream() {

    if (raami != null) {
        raami.release();
        raami = null;
    }

    virta.start();

}

private void paivitaKuva(ActionEvent ae) {
    if (raami != null) {
        raami.release();
        raami = null;
    }
    raami = virta.readFrame();
    ByteBuffer kuvaData = raami.getData().order(ByteOrder.LITTLE_ENDIAN);

    korkeus = raami.getHeight();
    leveys = raami.getWidth();
    korkeus2 = korkeus / puoli;
```

```
    leveys2 = leveys / puoli;

    if (kuva == null || kuva.length < leveys * korkeus) {
        kuva = new int[leveys * korkeus];
    }

    kuva2 = new int[leveys2][korkeus2];
    piirtokuva = new int[leveys2 * korkeus2];


    if (bKuva == null || bKuva.getWidth() != leveys || bKuva.getHeight() != korkeus) {
        bKuva = new BufferedImage(leveys, korkeus, BufferedImage.TYPE_INT_RGB);
    }
    if (lahimmat == null || lahimmat.length < herkkyys) {
        lahimmat = new int[herkkyys][2];
    }
    if (merkki < herkkyys - 1) {
        merkki++;
    } else {
        merkki = 0;
    }
    int i = 0;
    while (kuvaData.remaining() > 0) {
        int pix = kuvaData.getShort();
        kuva[i] = pix;
        i++;
    }

    laskePienin();
    bKuva.setRGB(0, 0, leveys2, korkeus2, piirtokuva, 0, leveys2);

    if (raami != null) {
        repaint();
    }
    raami.release();

}

private void laskePienin() {
    int pienin = 0xffff, lahi, posl = 0, posk = 0;
    float summax = 0, summay = 0;
    for (int y = 0; y < korkeus; y++) {
        for (int x = 0; x < leveys; x++) {
            lahi = kuva[x + y * leveys];
            if (lahi < pienin && lahi > 1) {
                pienin = kuva[x + y * leveys];
                posl = x;
                posk = y;
            }
```

```
            }
        }
        lahimmat[merkki][0] = posl;
        lahimmat[merkki][1] = posk;
        for (int i = 0; i < herkkyys; i++) {
            summax += lahimmat[i][0];
            summay += lahimmat[i][1];
        }
        lahinx = (int) (summax / herkkyys);
        lahiny = (int) (summay / herkkyys);
        lahinz = pienin;

        for (int y = 0; y < korkeus; y += puoli) {
            for (int x = 0; x < leveys; x += puoli) {
                kuva2[x / puoli][y / puoli] = kuva[x + y * leveys];
            }
        }
        for (int y = 0; y < korkeus2; y++) {
            for (int x = 0; x < leveys2; x++) {
                if (ui1) {
                    piirtokuva[x + y * leveys2] = (kuva2[x][y] << 8 | (kuva2[x][y] << 16) | (kuva2[x][y] << 24) | (kuva2[x][y] <<
32));
                } else {
                    piirtokuva[x + y * leveys2] = (kuva2[x][y]);//<<16|(kuva2[x][y]<<32)//|(kuva2[x][y]<<23)|(kuva2[x][y]<<31));
                }
            }
        }
    }

    protected synchronized void lopeta() {
        ajastin.stop();
        if (raami != null) {
            raami.release();
            raami = null;
            virta.stop();
            virta.destroy();
            xtion.close();
            OpenNI.shutdown();
        }
    }

    protected void teePuoli() {

        palkkix = 40 / puoli;
        palkkiy = 20 / puoli;
        palkkik = 30 / puoli;
        palkkiv = 60 / puoli;
    }
```

```
}
```

Kuva-luokka

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.*;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.util.ArrayList;
import javax.swing.JComponent;
import javax.swing.Timer;
import java.util.List;

import org.openni.*;

public class Nayttaja extends JComponent {

    private Device xtion;
    protected VideoStream virta;
    private VideoFrameRef raami;
    protected VideoMode mode;
    private int[] kuva, piirtokuva;
    private int[][] kuva2;
    private int leveys, korkeus, leveys2, korkeus2;
    protected int lahinx, lahiny, lahinz;
    protected BufferedImage bKuva;
    private ArrayList<SensorType> laiteSensorit;
    private PixelFormat pxf = PixelFormat.DEPTH_1_MM;
    protected int asetus;
    protected Color vari = new Color(0xff, 0xff, 0xff, 0xff);
    protected int herkkyys = 10, herkkyysz = 10;
    private int[][] lahimmat;
    private int merkki = 0;
    protected int puoli = 2;
    private int kleveys = 640, kkorkeus = 480, fps = 30, palkkix = 40 / puoli, palkkiy = 20 / puoli, palkkik = 55 / puoli, palk-
kiv = 60 / puoli;
    protected boolean ui1 = true;
    private int viive = 40;
    Timer ajastin = new Timer(viive, new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {
            paivitaKuva(ae);
        }
    });

    public Nayttaja() {
```

```
    List<DeviceInfo> devicesInfo = OpenNI.enumerateDevices();
    xtion = Device.open(devicesInfo.get(0).getUri());
    laiteSensorit = new ArrayList<>();

    if (xtion.getSensorInfo(SensorType.DEPTH) != null) {
        laiteSensorit.add(SensorType.DEPTH);
    }
    SensorType type = laiteSensorit.get(0);

    virta = VideoStream.create(xtion, type);
    java.util.List<VideoMode> supportedModes = virta.getSensorInfo().getSupportedVideoModes();
    mode = supportedModes.get(0);
    mode.setResolution(kleveys, kkorkeus);
    mode.setPixelFormat(pxf);
    mode.setFps(fps);
    virta.setVideoMode(mode);

    ajastin.start();

}

public synchronized void paintComponent(Graphics g) {

    if (raami == null) {
        return;
    }
    g.drawImage(bKuva, 0, 0, this);
    Graphics2D g2 = (Graphics2D) g.create();

    Rectangle alue = new Rectangle(512 / puoli, 384 / puoli);
    alue.setLocation(63 / puoli, 95 / puoli);
    g2.setColor(Color.red);
    g2.draw(alue);

    Rectangle sininen = new Rectangle(palkkik, palkkik);
    sininen.setLocation(palkkix, palkkiy);
    g2.setColor(Color.blue);
    g2.fill(sininen);
    g2.draw(sininen);

    Rectangle punainen = new Rectangle(palkkik, palkkik);
    punainen.setLocation(palkkix + palkkiv, palkkiy);
    g2.setColor(Color.red);
    g2.fill(punainen);
    g2.draw(punainen);

    Rectangle keltainen = new Rectangle(palkkik, palkkik);
    keltainen.setLocation(palkkix + palkkiv * 2, palkkiy);
```

```
g2.setColor(Color.yellow);
g2.fill(keltainen);
g2.draw(keltainen);

Rectangle vihrea = new Rectangle(palkkik, palkkik);
vihrea.setLocation(palkkix + palkkiv * 3, palkkiy);
g2.setColor(Color.green);
g2.fill(vihrea);
g2.draw(vihrea);

Rectangle musta = new Rectangle(palkkik, palkkik);
musta.setLocation(palkkix + palkkiv * 5, palkkiy);
g2.setColor(Color.black);
g2.fill(musta);
g2.draw(musta);

Rectangle valkoinen = new Rectangle(palkkik, palkkik);
valkoinen.setLocation(palkkix + palkkiv * 6, palkkiy);
g2.setColor(Color.white);
g2.fill(valkoinen);
g2.draw(valkoinen);

Rectangle magenta = new Rectangle(palkkik, palkkik);
magenta.setLocation(palkkix + palkkiv * 7, palkkiy);
g2.setColor(Color.magenta);
g2.fill(magenta);
g2.draw(magenta);

Rectangle syaani = new Rectangle(palkkik, palkkik);
syaani.setLocation(palkkix + palkkiv * 8, palkkiy);
g2.setColor(Color.cyan);
g2.fill(syaani);
g2.draw(syaani);

Rectangle oranssi = new Rectangle(palkkik, palkkik);
oranssi.setLocation(palkkix + palkkiv * 9, palkkiy);
g2.setColor(Color.orange);
g2.fill(oranssi);
g2.draw(oranssi);

Rectangle lopeta = new Rectangle(40 / puoli, 100 / puoli);
lopeta.setLocation(1, palkkiy * 13);
g2.setColor(Color.red);
g2.fill(lopeta);
g2.draw(lopeta);

Rectangle kohta = new Rectangle(5, 5);
kohta.setLocation(lahinx / puoli - 2, lahiny / puoli - 2);
g2.setColor(Color.red);
```

```
g2.fill(kohta);
g2.draw(kohta);

Rectangle aseta = new Rectangle(palkkik, palkkik);
aseta.setLocation(palkkix + palkkiv * 4, palkkiy);
g2.setColor(Color.red);
g2.draw(aseta);

Rectangle valittu = new Rectangle(palkkik + 15, palkkik + 15);
valittu.setLocation(palkkix + palkkiv * 9, palkkiy * 20);
g2.setColor(vari);
g2.fill(valittu);
g2.draw(valittu);

if (kohta.intersects(aseta)) {
   asetus = lahinz;
}

if (kohta.intersects(sininen)) {
   vari = Color.blue;
}
if (kohta.intersects(punainen)) {
   vari = Color.red;

}
if (kohta.intersects(keltainen)) {
   vari = Color.yellow;
}
if (kohta.intersects(vihrea)) {
   vari = Color.green;
}
if (kohta.intersects(musta)) {
   vari = Color.black;
}
if (kohta.intersects(valkoinen)) {
   vari = Color.white;
}
if (kohta.intersects(magenta)) {
   vari = Color.magenta;
}
if (kohta.intersects(syaani)) {
   vari = Color.cyan;
}
if (kohta.intersects(oranssi)) {
   vari = Color.orange;
}
if (kohta.intersects(lopeta)) {
   ajastin.stop();
   asetus = 0;
```

```
    }
}

protected void setStream() {

    if (raami != null) {
        raami.release();
        raami = null;
    }

    virta.start();

}

private void paivitaKuva(ActionEvent ae) {
    if (raami != null) {
        raami.release();
        raami = null;
    }
    raami = virta.readFrame();
    ByteBuffer kuvaData = raami.getData().order(ByteOrder.LITTLE_ENDIAN);

    korkeus = raami.getHeight();
    leveys = raami.getWidth();
    korkeus2 = korkeus / puoli;
    leveys2 = leveys / puoli;

    if (kuva == null || kuva.length < leveys * korkeus) {
        kuva = new int[leveys * korkeus];
    }

    kuva2 = new int[leveys2][korkeus2];
    piirtokuva = new int[leveys2 * korkeus2];


    if (bKuva == null || bKuva.getWidth() != leveys || bKuva.getHeight() != korkeus) {
        bKuva = new BufferedImage(leveys, korkeus, BufferedImage.TYPE_INT_RGB);
    }
    if (lahimmat == null || lahimmat.length < herkkyys) {
        lahimmat = new int[herkkyys][2];
    }
    if (merkki < herkkyys - 1) {
        merkki++;
    } else {
        merkki = 0;
    }
    int i = 0;
    while (kuvaData.remaining() > 0) {
        int pix = kuvaData.getShort();
```

```
            kuva[i] = pix;
            i++;
        }

        laskePienin();
        bKuva.setRGB(0, 0, leveys2, korkeus2, piirtokuva, 0, leveys2);

        if (raami != null) {
            repaint();
        }
        raami.release();

    }

    private void laskePienin() {
        int pienin = 0xffff, lahi, posl = 0, posk = 0;
        float summax = 0, summay = 0;
        for (int y = 0; y < korkeus; y++) {
            for (int x = 0; x < leveys; x++) {
                lahi = kuva[x + y * leveys];
                if (lahi < pienin && lahi > 1) {
                    pienin = kuva[x + y * leveys];
                    posl = x;
                    posk = y;
                }

            }
        }
        lahimmat[merkki][0] = posl;
        lahimmat[merkki][1] = posk;
        for (int i = 0; i < herkkyys; i++) {
            summax += lahimmat[i][0];
            summay += lahimmat[i][1];
        }
        lahinx = (int) (summax / herkkyys);
        lahiny = (int) (summay / herkkyys);
        lahinz = pienin;

        for (int y = 0; y < korkeus; y += puoli) {
            for (int x = 0; x < leveys; x += puoli) {
                kuva2[x / puoli][y / puoli] = kuva[x + y * leveys];
            }
        }
        for (int y = 0; y < korkeus2; y++) {
            for (int x = 0; x < leveys2; x++) {
                if (ui1) {
                    piirtokuva[x + y * leveys2] = (kuva2[x][y] << 8 | (kuva2[x][y] << 16) | (kuva2[x][y] << 24) | (kuva2[x][y] <<
32));
                } else {
```

```
                piirtokuva[x + y * leveys2] = (kuva2[x][y]);//<<16|(kuva2[x][y]<<32)//|(kuva2[x][y]<<23)|(kuva2[x][y]<<31));
            }
        }
    }
}

protected synchronized void lopeta() {
    ajastin.stop();
    if (raami != null) {
        raami.release();
        raami = null;
        virta.stop();
        virta.destroy();
        xtion.close();
        OpenNI.shutdown();
    }
}

protected void teePuoli() {

    palkkix = 40 / puoli;
    palkkiy = 20 / puoli;
    palkkik = 30 / puoli;
    palkkiv = 60 / puoli;
}
}
```

Lopeta-luokka

```
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class Lopeta extends WindowAdapter {

    Nayttaja na;

    Lopeta(Nayttaja n) {
        this.na = n;
    }

    public void windowClosing(WindowEvent e) {

        na.lopeta();
        System.exit(0);
    }
```