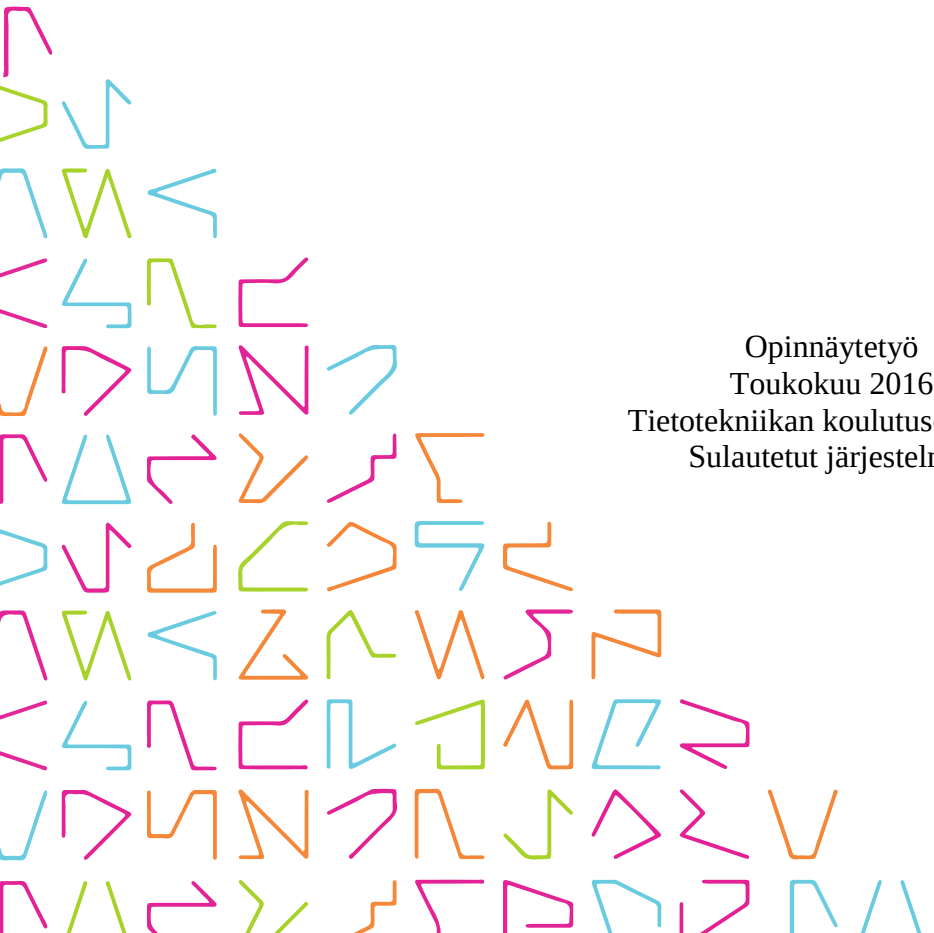
TAMPEREEN
AMMATTIKORKEAKOULU

KINECT SULAUTETUISSA JÄRJESTELMISSÄ

Olli Helminen



Opinnäytetyö
Toukokuu 2016
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Sulautetut järjestelmät

Olli Helminen:
Kinect sulautetuissa järjestelmissä

Opinnäytetyö 30 sivua, joista liitteitä 0 sivua
Toukokuu 2016

Kinect on Microsoftin Xboxille luoma laite, jonka pääasiallinen tarkoitus on pelien pelaaminen ihmisen kehon avulla, mutta sen edistykselliset ominaisuudet suhteessa sen myyntihintaan ovat tehneet siitä halutun sensorin sulautettujen järjestelmien projekteissa. Kinectin ominaisuuksiin kuuluvat videokuvan lisäksi millimetrin tarkkuuteen pystyvä syvyyskuva, infrapunakamera, liikkeiden tunnistusominaisuudet, puheen tunnistaminen erillisen kirjaston avulla sekä ihmisen kehon tunnistaminen.

Kinectin käyttämiseen tarvitaan Microsoftin tarjoama kirjasto jonka asentamalla tulee samalla asennettua Kinectin tarvitsemat ajurit ja kirjastot. Tietokoneessa täytyy myös olla USB 3 -portti, sillä Kinect vaatii toimiakseen USB 3 -väylän datansiirtoon. Kirjaston avulla voidaan hallita Kinectin ominaisuuksia pienen tutustumisen jälkeen ja Microsoft tarjoaa siihen liittyen suhteellisen hyvän kuvauksen eri funktiosta ja niiden toiminnasta. Kuinka niitä lopulta käytetään jää pelkkien kuvausten tutkimisella hankalaksi ja tarkoituksena onkin helpottaa Kinectin käyttöönottamista sulautettujen järjestelmien projekteissa.

Microsoft on myös luonut työkaluja omien eleiden luomista varten. Visual Gesture Builder -työkalun ja Kinect-studio nimisillä ohjelmilla voidaan luoda omia elekirjastoja joilla helpotetaan eleiden käyttämistä ja niiden prosessointia ohjelmoinnissa. Ei tarvita erikseen itse kirjoitettua koodia eleiden tunnistamiseen kun ne voidaan nauhoittaa ja opettaa kirjastolle jota voidaan referoida koodissa kun halutaan vastaanottaa eleitä.

Kinect on käytössä jo erilaisissa projekteissa. DARPA käyttää Kinectiä posttraumaattisen stressireaktion tunnistamiseen ja diagnosointiin sotilailla sekä NASA esimerkiksi on demonstroinut Kinectin käyttöä robotteja ohjattaessa. Kinectiä voidaan myös käyttää perinteisenä konenäkönä vaikkapa roboteissa.

Asiasanat: Kinect, käyttöönotto

ABSTRACT

Tampere University of Applied Sciences
Information technology
Embedded systems

Olli Helminen:
Kinect in embedded systems

Bachelor's thesis 30 pages, appendices 0 pages
May 2016

Kinect is device created for Xbox by Microsoft which is mainly used in gaming with human body but it also has some good technology inside compared to its price which has made it sought out device in embedded systems projects. Kinect houses full HD video camera, depth sensor capable to millimeter accuracy, infra-red camera, gesture recognizing capabilities, speech recognizing capabilities with additional libraries and human skeleton recognizing.

In order to use Kinect, you will need to install libraries and drivers provided by Microsoft. Your computer needs to also have USB 3 port to handle the data stream from Kinect. With the library you can control all the Kinect features with little practice. Microsoft also supplies you with quite good reference to all the functions inside Kinect library. Using all those functions only with this reference can be quite tricky so it's one of the reasons for this thesis to make using Kinect more accessible and give a good start to developing embedded systems with Kinect.

Microsoft also offers tools to create your own gestures. With VGB and Kinect-studio programs you can create your own gestures to a library which in turn make using gestures easier and processing them faster. You don't need to write your own solutions to recognize a gesture and you can instead record them on video and train your library with them. You can then use this library to recognize gestures in your code.

Kinect is already used in many projects around the world. For example DARPA is using Kinect to recognize post-traumatic stress disorder in soldiers and NASA has demonstrated how to use Kinect for controlling robots. You can also use Kinect as computer vision in robots.

Key words: Kinect, Tutorial

SISÄLLYS

LYHENTEET JA TERMIT

SDK	Software development kit
API	Application programming interface
MSFR	MultiSourceFrameReader
SDL	Simple DirectMedia Layer
VGB	Visual Gesture Builder

1 JOHDANTO

Työssä selvitetään Kinectin käyttömahdollisuuksia sulautetuissa järjestelmissä sekä käydään läpi esimerkkejä, joissa Kinectiä on käytetty esimerkiksi robotiikassa tai vaikkapa interaktiivisen näyttelyn luomiseen. Tutustutaan Kinectin ominaisuuksiin ja kuinka osaa niistä käytetään C++ -ohjelmointikielen avulla. Tutkitaan myös mitä muuta tarvitaan Kinectin datan saamiseksi näytölle ja luodaan perusta mahdolliselle jatkokehitykselle kun perusteet ovat hallinnassa.

Samalla käydään läpi SDL- ja OpenGL-kirjastojen alkeet, joita käytetään ikkunoiden luomiseen ja Kinectin datan visualisoimiseen näytölle. Selvitetään kuinka kirjastoja käytetään ja mitä tulee ottaa huomioon niitä käytettäessä, sekä tutkitaan funktioiden merkitystä ohjelman toimimisen kannalta.

Opetellaan käyttämään VGB:tä ja Kinect-studiota sillä niiden käyttäminen Microsoftin omien ohjeiden perusteella vaikutti vaikealta ja vaati hieman omakohtaista kokeilua. Luodaan esimerkkinä vaihe vaiheelta elekirjasto, jota voi sitten käyttää jatkossa Kinectin kanssa.

2 KINECT

Kinect on Microsoftin pelikonsolille kehittämä laite, jonka tarkoituksena oli mahdollistaa pelien pelaaminen ihmisen vartaloa käyttäen. Kinect ei kuitenkaan koskaan ole saavuttanut huimaa suosiota Xbox-käyttäjien keskuudessa joten se on suurelle osalle tuntematon laite. Erinäiset robotiikka kehittäjät ja jotkin yliopistot kuitenkin kiinnostuivat laitteen ominaisuuksista, koska se oli halpa laite verrattuna tekniikkaan jota se sisälsi. Tästä johtuen Kinect on suosittu laite konenäön luomista varten.

KUVA 1. Kinect 2.0 (Kinect for windows product blog)

Kinectin ensimmäinen versio ei kuitenkaan ollut Microsoftille riittävä, joten markkinoille tuotiin uusi Kinect 2.0-versio uuden Xbox-version kanssa. Tässä paremmassa versiossa oli parannuksia edelliseen nähden laajempi katselukulma syvyyskuvassa, syvyyskuva oli luotettavampi jopa puolen metrin päähän kamerasta. Videon resoluutio myös nostettiin 1080p HD-kuvan tasolle ja laitteella oli mahdollista seurata useampia vartaloita yhtäaikaisesti. Tässä työssä käytössä on Kinectin uudempi versio.

2.1 Ominaisuudet

Kinectissä on monia ominaisuuksia joita voidaan käyttää hyväksi erilaisten asioiden toteuttamiseen. Näihin ominaisuuksiin kuuluvat esimerkiksi videokuva, syvyyden määrittäminen, ihmisen tunnistaminen, kasvojen tunnistus, puheentunnistus sekä mm. eleiden tunnistaminen.

Kinectin liittämiseksi tietokoneeseen tarvitaan erillinen adapteri tai sellainen tulee itse rakentaa, jos ei voida käyttää verkkovirtaa. Kinect yhdistetään tietokoneeseen USB 3.0 -kaapelilla ja se vaati vähintään USB 3 -portin tietokoneesta toimiakseen. Energiaa Kinect kuluttaa joidenkin tietojen mukaan 16 W tunnissa, mutta tästä ei ole mitään varmaa lähdettä. Adapterin tiedoissa maksimiksi määritellään 12 V ja 2.67 A.

2.1.1 Kameran kuva

Kinectistä saadaan ulos normaalia värikuvaa HD-asetuksilla. Kameran kuvien resoluutio on 1920 x 1980 pikseliä ja se ottaa niitä 30 Hz taajuudella. Hämärissä olosuhteissa kuvien ottotaajuus laskee 15 Hz lukemiin.

2.1.2 Syvyyskuva

Kinectin syvyyttä mittaava kamera toimii infrapunalla ja sen varmaksi alueeksi, jolla tarkkuus on parhaimmillaan, on määritelty puolesta metristä aina neljään ja puoleen metriin asti. Tämän alueen ulkopuolelta voidaan saada myös syvyyslukemia, mutta ne eivät ole välttämättä tarkkoja. Syvyyskuvan resoluutio on 512x424 pikseliä ja sen kuvanottotaajuus on 30 Hz. Kameran katselukulma on 70 x 60 astetta. (Kinect Hardware 2016)

2.1.3 Kasvojen tunnistus

Kinectin kasvojen tunnistus on hyvin korkealla tasolla. Se pystyy tunnistamaan ihmisen kasvoista ilmeitä, ovatko silmät auki vai ei, onko ihminen keskittynyt Kinectiin vai onko hänen huomionsa kiinnittynyt muualle. Sillä pystyy tunnistamaan päänasennon myös muuhun kehoon nähden.

Kasvojen tunnistamisessa vaikuttavia tekijöitä ovat päänasento Kinectiin nähden, mahdollinen lika Kinectin kameroiden edessä ja valaistuksen taso kohteen kasvoilla. Myös kasvojen etäisyys vaikuttaa Kinectin kasvojen tunnistamiseen. Liian lähellä tai kaukana olevat kasvot eivät ole tunnistettavissa ja niistä ei saada edellä mainittuja tietoja luettua.

2.1.4 Vartaloiden tunnistus

Kinectissä on ominaisuutena kyky tunnistaa ihmisen vartalo sen kuvaamasta alueesta ja todeta missä asennossa ihmisen raajat ovat. Se pystyy seuraamaan yhtä aikaa kuutta eri vartaloa ja jokaisessa vartalossa 25 eri raajaa ja niiden tilaa. Kinect pystyy myös erottelemaan onko ihmisen kädet auki vai kiinni tai tietynlaisessa ”lasso” tilassa jossa henkilön etu- ja keskisormi ovat pystyssä ja muut ovat kiinni. Tätä voi käyttää esimerkiksi osoittamiseen.

Seuratut vartalot ovat orientoitu niin kuin ne katsoisivat peiliin, joten ohjelmien kehittämisen yhteydessä kameran oikealla puolella oleva käsi on myös koodissa oikean puoleinen käsi. Vartalon tunnistamisessa on ominaisuutena myös kuinka paljon kohdevartalo nojaa poispäin normaalista pystyasennosta. Tämä on mahdollista havaita niin sivuttais kuin pitkittäis suunnassa.

2.1.5 Puheentunnistus

Kinect pystyy tunnistamaan mistä suunnasta sen mikrofoniin tulee ääntä ja käsittelemään tätä ääntä. Varsinaisia puheentunnistus ominaisuuksia siinä itsessään ei ole vaan ne täytyy luoda jollakin sitä varten tarkoitetulla kirjastolla erikseen. Ominaisuuden avulla voidaan kuitenkin luoda sovelluksia joissa Kinect kiinnittää huomionsa ääniin jotka tulevat esimerkiksi joltain ennalta määritellyltä sektorilta.

2.2 SOFTWARE DEVELOPMENT KIT

Microsoft toimittaa Kinectille valmiina SDK:n jossa on kaikki tarvittavat elementit joita Kinectin kanssa tarvitaan. SDK:n sisällä on määriteltynä funktiot ja oliot kaikille sen ominaisuuksille, joten ohjeistusta tutkimalla voidaan kirjoittaa koodia jota käyttäen saadaan Kinectistä kaikki ominaisuudet käyttöön. Tärkeimmät ominaisuudet sulautettuja järjestelmiä ajatellen tulevat varmasti olemaan syvyyskuvan ja siitä saatavan datan käsittely, mutta varmasti muistakin ominaisuuksista on järjestelmästä projektista riippuen hyötyä.

Jotta SDK:ta voidaan käyttää, tarvitaan Kinectin oman kirjaston lisäksi muutamia muita kirjastoja. Näiden tarve johtuu täysin siitä, että Kinect-kirjasto käyttää näitä kirjastoja omissa funktioissaan joten niiden täytyy olla mukana koodia kirjoitettaessa. Kinect-kirjaston tarvitsemat kirjastot ovat Windows.h ja Ole2.h. Windows-kirjasto pitää sisällään kaikki Windowsin API:n funktioiden esittelyt ja Ole2.h sisältää tarvittavia funktioita datan siirtämiseen eri applikaatioiden välillä.

2.2.1 Kinectin alustaminen

Jotta Kinectiä voitaisiin käyttää ohjelmassa, on se aluksi alustettava käyttövalmiiksi ohjelman alussa. Aluksi määritellään IKinectsensor-tyyppinen olio jota käytetään sitten sensorin hallintaan. Tässä tapauksessa luodaan olio jonka nimi on sensor. Seuraavaksi alustuksessa varmistetaan, että tietokoneeseen on kytkettynä Kinect-sensori kutsumalla funktiota jolla haetaan oletussensori ja annetaan sille attribuutiksi luotu olio. Jos löydettiin Kinect-sensori, se valjastetaan sensor-olion käyttöön. Alustus lopetetaan jos getdefaultkinectsensor-metodi palauttaa virheen.

```
bool initKinect()
{
    if (FAILED(GetDefaultKinectSensor(&sensor)))  //get default kinect sensor. If its not found step out of the init
    {
        return false;
    }
    if (sensor)
    {
        sensor->Open(); //open data streams on kinect

        sensor->get_ColorFrameSource(&colorFramesource); //get colorframesource
        colorFramesource->OpenReader(&colorReader);     //open reader so that we can read colorframes
        if (colorFramesource)                           //if there is frame waiting
        {
            colorFramesource->Release();                //release the frame and return null. Kinect is ready
            colorFramesource = NULL;
        }
        return true;
    }
    else
    {
        return false;
    }
}
```

KUVA 2. Kinectin init-funktio (Edward Zhang)

Kun on varmistuttu siitä, että laitteessa on kiinni Kinect-sensori ja sille on ohjaimet asennettuna voidaan siirtyä eteenpäin alustuksessa. Seuraavaksi täytyy käyttää sensori-oliota open-metodia jolla Kinect avataan käyttöä varten ja se voi vastaanottaa muita komentoja

sekä luovuttaa keräämäänsä dataa. Seuraavaksi haetaan sensor-oliolle tieto missä kameran ottamat kuvat sijaitsevat muistissa. Tämä tapahtuu kutsumalla get_colourframesource-funktiota jolle annetaan attribuutiksi kyseisen muuttujan osoite johon voidaan tallentaa paikka muistissa, josta kuvia noudetaan muun koodin käytettäväksi. Tämän jälkeen luodaan vielä kuvien lukija jonka avulla kuvat saadaan lopullisesti ulos Kinectistä. Lopuksi varmistetaan vielä, että colorframesourcessa on kuva odottamassa ja jos näin on, vapautetaan se jolloin voidaan todeta Kinectin olevan valmis videokuvan osalta.

Muut ominaisuudet alustetaan samalla tavalla. Haetaan jokaiselle tarvittavalle ominaisuudelle source ja tehdään sille lukija. Tämän jälkeen ne ovat käytettävissä ohjelmassa ja niistä saadaan dataa ulos.

2.2.2 Värikuvan käsittely

Kun Kinect on alustettu onnistuneesti, voidaan sieltä alkaa pyytämään dataa. Tämä tapahtuu luomalla Icolorframe-olio jota koodissa kutsutaan frameksi. Tämä olio edustaa siis Colorframesourcesta saatua kuvaa, jotta sitä voidaan käsitellä.

```
void GetKinectColor(GLubyte* dest)
{
    IColorFrame* frame = NULL; //create place to store the frame
    if (SUCCEEDED(colorReader->AcquireLatestFrame(&frame))) // if getting frame from kinect succeeded
    {
        frame->CopyConvertedFrameDataToArray(width*height * 4, data, ColorImageFormat_Bgra);
    }//copy raw data from kinect to array and convert it to BGRA

    if (frame) frame->Release(); //release frame so that new one can be fetched later
}
```

KUVA 3. Kinect värikuvan käsittely funktio (Edward Zhang)

ColorReaderin metodilla haetaan viimeisin saatu kuva ja tallennetaan se juuri luotuun olioon. Tämän jälkeen muunnetaan sensorilta saatu raakakuva joksikin yleisemmäksi kuvaformaatiksi, tässä tapauksessa BGRA-formaattiin. Tämä tapahtuu colorframe luokan metodilla CopyConvertedFrameDataToArray, joka muuttaa raakakuvan haluttuun kuvaformaattiin ja tallentaa sen osoitettuun taulukkomuuttujaan. Tämän jälkeen kuvan voidaan sitten taulukosta tulostaa ikkunaan halutulla metodilla. Lopuksi vielä vapautetaan framesta kuva, jotta voidaan ottaa jatkossa seuraava taas käsittelyyn.

2.2.3 Syvyysdatan käsittely

Syvyyskuvan datan saaminen muun koodiin käytettäväksi toimii samoin kuin kuvan 3 esimerkissä, vaihdetaan vain kaikki color-sanat depth-sanaan. Täytyy myös muistaa käyttää framea käsiteltäessä syvyyskuvan 512x424 resoluutiota. Tämän jälkeen voidaan samoilla metodeilla pyytää Kinectiltä syvyyskuvia. Tällä kertaa kuitenkin Kinect palauttaa vastauksena taulukon jossa jokaisen alkion sisällä on 16-bittinen luku, joka kertoo millimetreinä kyseisen pikselin etäisyyden Kinectistä. Täten Kinectin maksimietäisyydeksi, joka voidaan mitata, saadaan noin 8 metriä, mutta luotettava tarkkuus loppuu kuitenkin 4,5 metrin jälkeen.

```
DepthFrame->CopyFrameDataToArray(512 * 424, DepthLevels);

if (DepthFrame) DepthFrame->Release();

do
{
    if (DepthLevels[counter] <= vertailu)
    {
        DepthImage[counter * 4] = 0;
        DepthImage[counter * 4+1] = 0;
        DepthImage[counter * 4+2] = 0;
        DepthImage[counter * 4+3] = 255;
    }
```

KUVA 4. Syvyysdatan käsittely

Data kopioidaan CopyFrameDataToArray-funktiolla DepthLevels-taulukkoon. Tätä taulukkoa ei voi suoraan vielä piirtää näytölle, sillä se sisältää vain syvyysdata millimetreinä. Tämän jälkeen käydään jokainen taulukon alkio läpi ja kun löydetään pikseli jonka etäisyys on pienempi tai yhtä suuri kuin vertailuluku, koodissa 1000 mm eli yksi metri, kirjoitetaan toiseen taulukkoon jokaisen värin kohdalle 0 ja alpha-bitiksi 255 jolloin pikseli ei ole läpinäkyvä. Jos etäisyys taas on yli metrin, pikseli värjätään valkoiseksi. Näin saadaan aikaan taulukko joka voidaan sitten piirtää OpenGL:n avulla kuvaksi.

2.2.4 Useamman lähteen yhtäaikainen käyttö

Jos joudutaan käyttämään useampaa Kinectin ominaisuutta yhtä aikaa, on helpompaa käyttää MultiSourceFramea sen sijaan, että käyttäisi jokaista lähdettä erikseen. Tämä tapahtuu käyttämällä IMultiSourceFramea, kuten käyttäisi muitakin lähteitä, mutta sensoria

avattaessa määritellään erikseen avattavaksi MultiSourceFrameReader ja mitä lukijoita sen kautta käytetään. Tätä käytettäessä tulee kuitenkin yksi lisävaihe. MSFR:stä haetaan ensin normaalisti frame josta sitten haetaan erikseen vielä tarkemmin mitä framea halutaan käsitellä. Tämä tapahtuu esimerkiksi syvyyskuvan kohdalla IDepthFrameReferencen avulla. Framesta haetaan referenssi syvyyskuvalle ja tallennetaan se syvyysreferenssiin. Tämän jälkeen tätä referenssioliota hyväksi käyttämällä haetaan uusin syvyyskuva jota sitten voidaan käsitellä normaalisti. Lopuksi tulee muistaa vapauttaa referenssi sekä normaalisti viimeisin haettu syvyyskuva.

```
void getColorData(IMultiSourceFrame* frame, GLubyte* dest)
{
    IColorFrame* ColorFrame; //create place to store the depth frame from kinect
    IColorFrameReference* Colorframeref = NULL; //create place for frameref from which the actual depthframe is fetched
    frame->get_ColorFrameReference(&Colorframeref);  //get the reference from multisource
    Colorframeref->AcquireFrame(&ColorFrame); //get the depth frame with the help of the reference
    if (Colorframeref) Colorframeref->Release(); //release reference
    if (!ColorFrame) return; //abort if depthframe was not found

    ColorFrame->CopyConvertedFrameDataToArray(width * height * 4, ColorImage, ColorImageFormat_Rgba);

    if (ColorFrame) ColorFrame->Release();
}
```

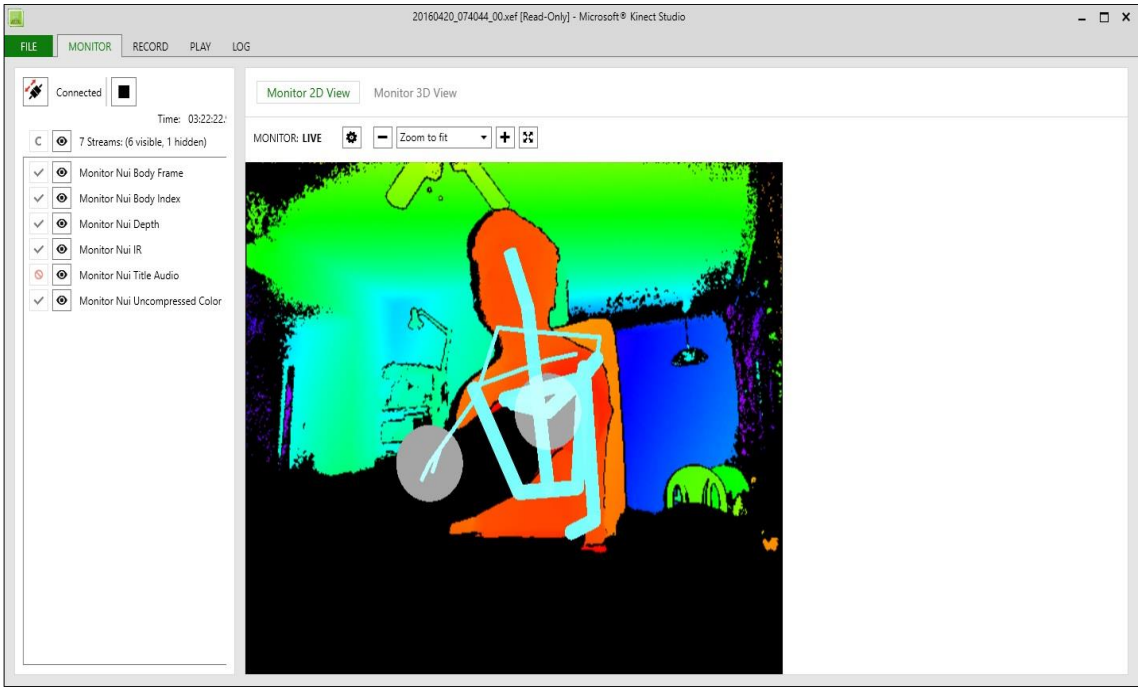
KUVA 5. Multisourceframen käyttäminen (Edward Zhang)

Kuvan kolme tapaista menettelyä voidaan käyttää myös syvyysdatan käsittelyyn. vaihdetaan vain kaikki color-sanat depth-sanoiksi ja muokataan hieman datan kopiointifunktion argumentteja vastaamaan syvyyskuvan parametreja.

2.2.5 VGB-Työkalu ja Kinect-studio

VGB eli Visual Gesture Builder on Microsoftin työkalu omien elekirjastojen luontiin. Työkalun avulla voidaan esimerkeistä tehdä valmiita malleja omaan kirjastoon, jota voidaan siten vertailla ohjelmien suorituksen aikana. Elekirjaston luomisen etuja on eleiden tunnistamisen varmuuden lisääminen, vältytään suurelta koodin kirjoittamisen määrältä ja nopeutetaan eleiden tunnistamista. VGB ja Kinect-studio ovat Kinectin SDK:n mukana tulevia työkaluja, joten niitä ei tarvitse erikseen ladata.

Kinect-studio on ohjelma jossa voidaan nauhoittaa videoklippejä, joita sitten käytetään VGB:ssä eleiden luontiin ja harjoittamaan niiden tunnistamista. On hyvä ottaa useita klippejä, sillä yksittäisen eleen toistaminen tarkalleen on lähes mahdotonta juuri niin kuin sen on kerran klipissä tehnyt. Tosin VGB:ssä voi säätää myöhemmin kuinka tarkasti kyseiset liikkeet tulee suorittaa.

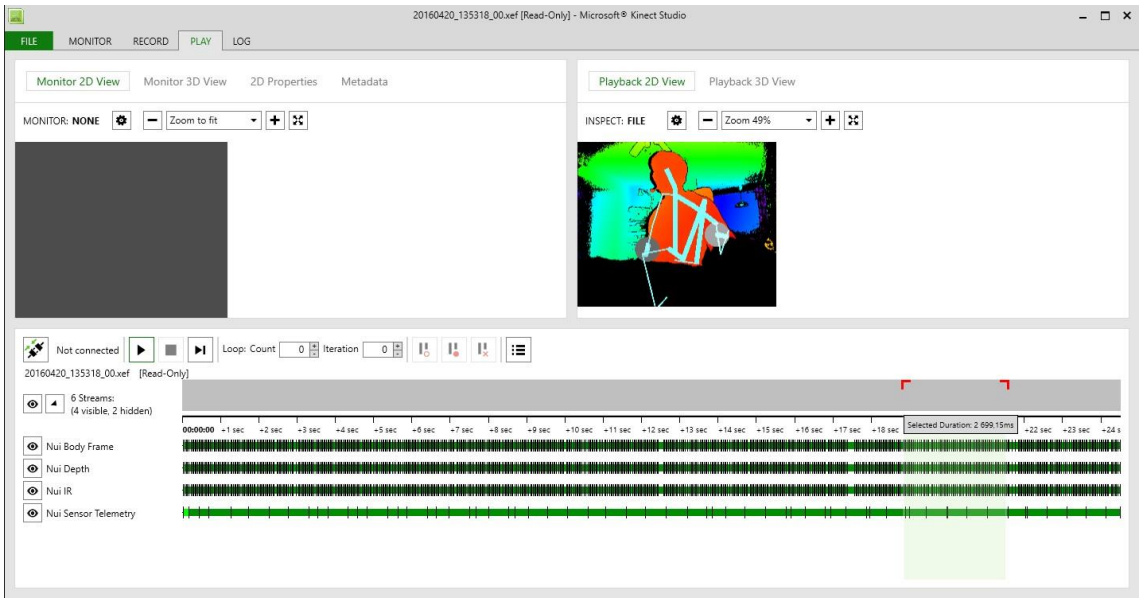


KUVA 6. Kinect Studio

Aloitetaan luomalla Kinect-studiossa muutama videoklippi, joita voidaan käyttää VGB:ssä. Ensiksi tulee yhdistää Kinect-studio Kinectiin ja se tapahtuu yksinkertaisesti painamalla nappia not connected-tekstin vieressä tai file-menun kautta connect to service. Tämän jälkeen tulisi vasemmalla ruudulla näkyä Kinectin syvyyskuva ja jos se on tunnistanut ihmisen kuvasta niin myös henkilön luuranko. Vasemmalla on listattuna kaikki Kinectistä saatavat streamit joista voidaan valita ne, joita halutaan tallentaa ja käsitellä. Kuvassa on nähtävissä myös pallot käsien kohdalla jotka vaihtavat väriä käden eri tilojen mukaan. Harmaa tarkoittaa määrittelemätöntä, punainen kiinni olevaa ja vihreä aukinaista kättä.

Kun on varmistuttu Kinectin toimivuudesta, voidaan siirtyä tallentamaan klippejä käyttöä varten. Tämä tapahtuu record-välilehdeltä jossa on sama kuva kuin monitor-sivullakin. Vasemmalla laidalla on jälleen lueteltuna kaikki mahdolliset streamit jotka voidaan tallentaa. Valitaan halutut streamit tallennettavaksi ja painetaan tallennusnappia kuvaamisen

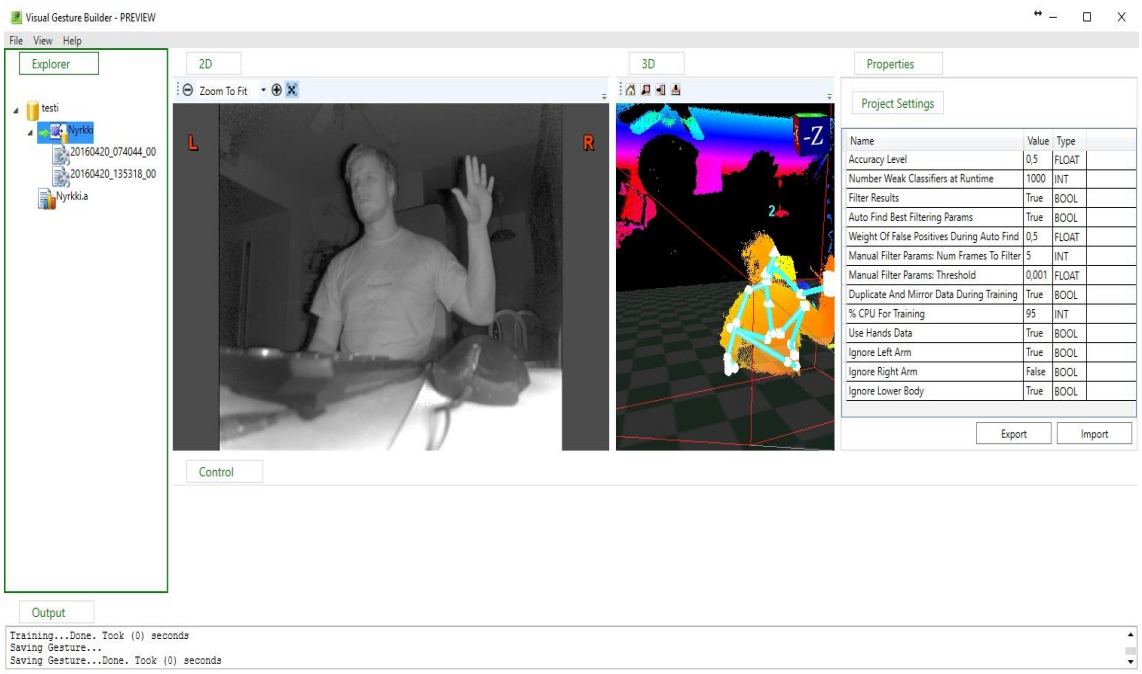
aloittamiseksi, toistetaan elevideolla joka halutaan lisätä kirjastoon ja lopetetaan tallennus. Kinect-studio automaattisesti tallentaa luodun xef-tiedoston jota tarvitaan myöhemmin VGB:ssä kun kirjastoon opetetaan ele. Kinect-studio myös automaattisesti pitää kaikkien streamien datan synkronoituna ajan suhteen.



KUVA 7. Kinect-studion play-ikkuna

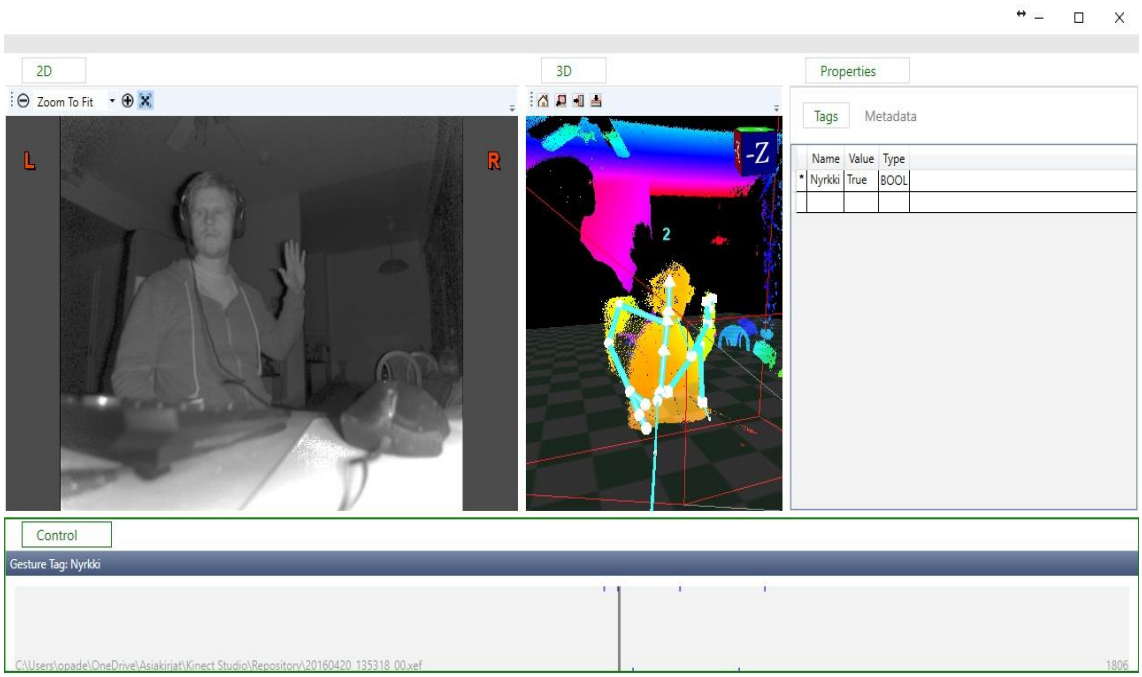
Play-välilehdeltä voidaan tarkastella sen hetkistä monitorin tilaa tai juuri äsken tallennettua videoklippiä. Alapuolella on aikajana jossa on eri streamien tallentamat tapahtumat ja sitä voi kelata hiirtä liikuttelemalla edestakaisin. Aikajanalta voidaan myös eritellä sen tietty osa jossa tehdään haluttua elettä ja rajata klippi pitämään sisällään vain rajattu ajanhetki. Tämä tapahtuu maalaamalla alue jonka haluat ja hiiren oikealla napilla valitsemalla set in/out points to selection range, jonka jälkeen rajatun alueen alkuun ja loppuun tulisi siirtyä punaiset sitä rajaavat merkit.

Kun on saatu äänitettyä muutama video jossa esiintyy haluttu ele, voidaan siirtyä VGB:n puolelle luomaan varsinaista kirjastoa. Aloitetaan luomalla ensimmäisenä uusi solution johon luodaan sitten uusi projekti. Tähän kannattaa käyttää ohjattua luomista jossa määritellään, mitä kaikkea eleessä tarvitaan ja tulee ottaa huomioon. Tässä tapauksessa eleen ollessa nyrkin aukaiseminen ja sulkeminen voidaan luomisen yhteydessä kertoa alaraajojen olevan merkityksettömiä. Samalla todetaan, että seurataan vain oikeanpuoleista kättä ja käden tilalla on merkitystä eleessä. Kun projekti on luotu, voidaan siihen liittää aikaisemmin Kinect-studiossa luotu klippi. Tämä tapahtuu valitsemalla hiiren oikealla projektin päältä add clip.



KUVA 8. VGB-ikkuna

Kuvassa kahdeksan nähdään VGB:n pääikkuna ja lisätyt klipit oikealla tiedostopuussa. Keskellä on esitettynä klippien sisältö johon voidaan valita näkyviin halutut streamit kyseisestä klipistä. Oikealla on taulukko jossa on esitettynä projektin luomisen yhteydessä asetetut arvot. Näistä on ehkä tarpeellista tiputtaa tarkkuus lukema 0.95:stä esimerkiksi 0.5:een. Tämä siksi, että kirjasto ei olisi turhan tarkka eleen täydellisestä onnistumisesta ja se sallisi pieniä epäkohtia. Kun arvot on säädetty halutuiksi ja klipit on lisätty projektiin, voidaan aloittaa eleen tunnistamisen räätälöinti.



KUVA 9. VGB eleen asettaminen

Kun haluttu klippi on valittuna, ilmestyy control kohtaan aikajana josta etsitään ajankohta jossa ele esiintyy. Tässä tapauksessa halutaan jotakin tapahtuvan kun oikea nyrkki on auki, joten kun osoitin on oikeassa kohdassa, asetetaan oikeanpuoleisessa sarakkeessa olevaan arvo kohtaan sana true. Jos klipissä on toistettu useamman kerran elettä, on hyvä näitä arvoja laittaa myös muihin kohtiin jossa nyrkki on auki asennossa. Myös kohtiin jossa nyrkki on kiinni, on hyvä laittaa false merkkejä, jotta eri tilojen erottelu olisi helpompaa. Kun on asetettu tarpeeksi merkkejä, valitaan projektin kohdalta build jolloin VGB rakentaa kirjaston.

KUVA 10. Eleen testaus

Kun VGB on onnistuneesti luonut kirjaston, sitä täytyy testata. Tämä tapahtuu klikkaamalla projektin päältä ja valitsemalla live preview-vaihtoehto. Kohta pitäisi avautua kuvan 10 mukainen ikkuna jossa vasemmalla näkyy mustavalkoinen syvyyskuva luurangon kanssa ja tietoja testin tilasta. Oikealla olevassa laatikossa näkyy pystyviivoina kuinka varma ohjelma on siitä, toteutetaanko elettä kyseisellä hetkellä. Tässä tapauksessa jos nyrkki on auki eli tosi tulisi ruudun täyttyä pystyviivoista ja nyrkin ollessa suljettuna pitäisi viivojen kadota kokonaan. Viivojen korkeus kuvaa ohjelman varmuutta siitä kuinka hyvin se on mielestään tunnistanut sille opetetun eleen.

3 MUUT KIRJASTOT

3.1 Alkuvaikeudet

Työn aloittamisessa oli ensiksi vaikeutena kaikkien kirjastojen saaminen toimintaan yhtä aikaa. Windows- ja Ole2-kirjastot ovat natiivisti Windows ympäristössä, joten ne toimivat vain normaalisti lisäämällä niiden header tiedostot koodiin. Kinect-kirjaston ja tässä tapauksessa SDL-kirjaston toimintaan saaminen tuotti vaikeuksia joiden selvittelyyn meni aikaa.

Osasyynä kirjastojen lisäämisen vaikeuksissa varmasti näytteli Visual Studion kankeus kirjastojen lisäämisessä. Projekti jouduttiin luomaan useita kertoja uudestaan jotta se saatiin lopulta kääntymään.

3.1.1 Kirjastojen lisääminen projektiin

Kirjastot saatiin lopulta lisättyä projektiin tutkimalla erinäisiä ohjeita ympäri internetiä. DLL-tiedostot kirjastoista lisätään 64-bittisessä käyttöjärjestelmässä Windows kansion alla sijaitsevaan sysWOW64 kansioon system32 kansion sijasta. Tämän lisäksi täytyy käyttää 32-bittisiä kirjastoja Visual Studion kanssa, koska se 64-bittisestä versiosta huolimatta kääntää ohjelmat edelleen 32-bittisenä. Jos käytettiin 64-bittisiä kirjastoja, ei koodi kääntynyt odotetusti, vaan saatiin aikaiseksi vain linkkeri virheitä. Tämä virhe saatiin myös aikaiseksi vain Kinectin kirjastoilla, joten on mahdollista, että virhe on vain Kinectin 64-bittisessä kirjastossa.

DLL-tiedostojen lisäämisen lisäksi tulee Visual Studiolle osoittaa missä header-tiedostot ovat kyseisille kirjastoille. Tämän pitäisi olla hyvin yksinkertainen toimenpide, lisätään vain tiedostojen sijainti ja nimi projektin asetuksiin C++- ja linker-osiossa. Näin ei kuitenkaan ollut vaan jouduttiin tekemään kokonaan uusi projekti jos polku meni ensimmäisellä kerralla väärin. Tähän virheeseen ei ikinä löytynyt järkevää selitystä, vaan koodi lähti kääntymään kun polut saatiin kerralla asetettua oikeiksi.

SDL-kirjastojen lisääminen tapahtui samalla tavalla kuin Kinectinkin kirjastojen lisääminen. Tällä kertaa täytyi vain olla tarkkana, että käytettiin sama bittistä versiota Kinectin kanssa. Jos näin ei tehty, koodi kyllä kääntyi hienosti, mutta ohjelman suorittamisen alussa tuli virheilmoitus virhekoodin kera. Tästä päästiin eroon vaihtamalla SDL 64-bittinen kirjasto 32-bittiseen kirjastoon, jotta se olisi sama kuin käytettävä Kinect-kirjasto.

3.2 SDL-kirjasto

Työssä käytetään ikkunoiden luontiin ja hallintaan Simple DirectMedia Layer-kirjastoa. Kyseisen SDL-kirjaston avulla on helppo luoda ikkunat värikuvaa varten ja mahdollisten syvyyskuvan ja muiden ominaisuuksien debuggaamiseen.

3.2.1 SDL-kirjaston alustaminen

SDL alustetaan yksinkertaisesti kutsumalla SDL_init-funktiota, jolle annetaan argumentteina tarvittavat SDL:n ominaisuudet. Helpoin tapa on antaa argumenttina init everything, jolloin SDL on valmiina käyttämään kaikkia sen ominaisuuksia. Järkevän muistinkäytön kannalta olisi tietenkin parempi, jos käynnistettäisiin vain tarvittavat SDL:n osat, mutta työssä ei optimointi ole vielä sillä asteella, että olisi järkevä käyttää jotakin muuta init konfiguraatiota.

```
SDL_Init(SDL_INIT_EVERYTHING);
SDL_Window* screen =
    SDL_CreateWindow("Kinect", SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED, SCREEN_WIDTH, SCREEN_HEIGHT, SDL_WINDOW_OPENGL);
SDL_GLContext glcontext = SDL_GL_CreateContext(screen);
```

KUVA 11. SDL:n alustaminen

Kun kyseinen funktio on suoritettu, voidaan luoda ikkunoita tarpeen mukaan. Käytetään SDL_window-structurea, nimetään se ja kutsutaan SDL_CreateWindow-funktiota. Ensimmäisenä argumenttina annetaan ikkunan nimi, seuraavat argumentit ovat järjestyksessä, Ikkunan sijainnin X koordinaatti, Ikkunan sijainnin Y koordinaatti, ikkunan leveys, ikkunan korkeus ja viimeisenä vielä mahdollisia lisä argumentteja. Tässä tapauksessa ikkunalle kerrotaan vielä, että sitä tullaan käyttämään OpenGL-kirjaston kanssa. Lopuksi

luodaan ikkunasta vielä OpenGL:ää varten konteksti, jotta siihen voidaan piirtää OpenGL:än avulla kuvia ja päivittää kyseistä kuvaa.

3.2.2 SDL:n käyttäminen

SDL:ää käytetään tämän jälkeen piirtämällä valmisteltavaan kuvaan OpenGL:n avulla uusi Kinectistä saatu kuva ja kun se on piirretty, vaihdetaan se näytettäväksi ikkunassa. Samalla kuunnellaan SDL:n mahdollisia tapahtumia joihin lukeutuu ikkunan sulkeutuminen. Tässä tapauksessa aina ennen uuden kuvan piirtämistä ja noutamista Kinectistä suoritetaan tarkastus onko SDL:n luoma ikkuna suljettu. Jos se on suljettu, SDL_event-muuttujassa ev on SDL_QUIT-lippu. Tällöin lopetetaan ohjelman suorittaminen. Uusien kuvien noutamista ja piirtämistä jatketaan kunnes käyttäjä sulkee ikkunan.

```
SDL_Event ev; //create SDL event structure named ev
bool running = true; // while loop controller
while (running) {
    while (SDL_PollEvent(&ev)) { //poll SDL event. If false make running false
        if (ev.type == SDL_QUIT) running = false;
    }
    drawKinectData(); //get kinect data and draw it with opengl
    SDL_GL_SwapWindow(screen); //swap new frame to screen
}
```

KUVA 12. SDL main loop (Edward Zhang)

Ikkunaan piirtäminen tapahtuu OpenGL:n avulla josta on seuraavaksi kerrottu enemmän. SDL:n avulla täytyy kuitenkin vaihtaa vanha kuva uuteen ja se tapahtuu käyttäen SDL:n funktiota SDL_GL_Swapwindow. Tällöin taustalle valmiiksi piirretty kuva vaihdetaan nykyisen kuvan tilalle näytettäväksi. Näin tehdään jotta ikkunassa ei näkyisi kun OpenGL piirtää kuvaa reaaliaikaisena vaan näytetään vain valmis lopputulos. (SDL reference 2016)

3.3 OpenGL-kirjasto

Työssä käytetään OpenGL-kirjastoa piirtämään kuvia SDL-ikkunoiden sisään. Jotta näin voitaisiin tehdä, joudutaan ensin alustamaan OpenGL:ssä tarvittavat attribuutit ja luomaan tekstuureille muistipaikat, josta niitä voidaan käyttää piirtoalustana Kinectistä saaduille kuville tai syvyyskuvan datan visualisointiin.

3.3.1 OpenGL:n alustaminen

OpenGL:n alustaminen ei olekaan niin yksinkertaista kuin muiden tarvittavien toimintojen. OpenGL:ssä joudutaan ensimmäisenä luomaan tekstuurille jokin identifikaatio numero, jotta jokaista tekstuuria olisi helpompi käyttää ja muokata ohjelmassa. Tässä tapauksessa, kuten kuvasta kuusi voidaan todeta, käytetty id numero tekstuurille on yksi. Tämän jälkeen tekstuurille on määriteltävä millainen tekstuuri se on. Esimerkissä tekstuurista tehdään kaksiulotteinen kuva ja se sidotaan kyseiseen tekstuuri kohteeseen. Se pysyy sidottuna kyseiseen GL_TEXTURE_2D-kohteeseen kunnes se erikseen irrotetaan siitä, tai sen tilalle asetetaan jokin muu tekstuuri käsiteltäväksi. Muita mahdollisia kohteita olisi esimerkiksi 3D- tai 1D-tekstuurit, mutta ne eivät tässä työssä ole tarpeellisia.

```
//Initialize textures
glGenTextures(1, &textureId);
glBindTexture(GL_TEXTURE_2D, textureId);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height,
    0, GL_BGRA, GL_UNSIGNED_BYTE, (GLvoid*)ColorImage);
```

```
// OpenGL setup
glClearColor(0.f, 0.f, 0.f, 0.f);
glClearDepth(1.0f);
glEnable(GL_TEXTURE_2D);
```

KUVA 13. OpenGL:n init-funktiot (Edward Zhang)

Seuraavaksi asetetaan kyseiselle tekstuurille parametreja. Tekstuurin pienennys funktiota käytetään jos tekstuuria luodessa huomataan tekstuurin tarkkuuden tarvitsevan pienentämistä. Funktiolle myös annetaan argumentiksi mitä funktiota sen tulisi käyttää kuudesta mahdollisesta pienentämisfunktiosta. Tässä tapauksessa parametriksi annetaan

GL_NEAREST jolloin käytetään lähintä tekstuuri elementtiä spesifioiduista koordinaateista.

Toisessa parametrin asetus funktiossa kerrotaan tekstuurille, mitä sen tulee tehdä jos tekstuuria luodessa sen tarkkuutta tulisi suurentaa. Samassa annetaan argumentiksi tällekin funktiolle, että käytetään lähintä tekstuuri elementtiä kyseisten koordinaattien läheisyydessä, jossa suurentamista tarvitaan.

gltextimage2D-funktiolla viimein asetetaan jotakin dataa tekstuuriin, tässä tapauksessa lisätään vain kuva joka sisältää alustuksen jälkeen taulukossa olevan datan. Kyseiselle funktiolle joudutaan kuitenkin antamaan kasa erinäköisiä argumentteja joita tarkastellaan seuraavaksi. Ensimmäisenä argumenttina funktio saa kohde tekstuurin johon kuva lisätään. Tässä tapauksessa kohde tekstuuri on GL_TEXTURE_2D johon aiemmin on kiinnitetty tekstuuri id yksi. Seuraavana argumenttina annetaan tarkkuus taso jota käytetään kyseisestä tekstuurista. Kun tämä arvo on asetettu nollaksi, käytetään kuvan perustasoa eli kuvaa sellaisenaan kun se tulee kamerasta. Jos käytettäisiin jotain muita parametreja tekstuuri parametrien asettamisen yhteydessä, tähän voisi silloin asettaa jonkin muun numeron joka vastaisia seuraava mipmap redusoitua kuvaa alkuperäisestä kuvasta.

Kolmantena argumenttina funktiolle kerrotaan mitä sisäistä formaattia sen tulee käyttää. Tällä kerrotaan funktiolle kuinka monta värikomponenttia kuvassa on ja missä järjestyksessä ne annetussa datassa ovat. On myös mahdollista käyttää tiettyä resoluutiota tekstuurien varastointiin funktiossa. Tämä tehdään asettamalla parametriksi jokin tiettyä kokoa osoittava sisäinen formaatti, jolloin funktio valitsee jonkin sitä läheisesti vastaavan sisäisen esitysmuodon, mutta se ei välttämättä ole täysin sama kuin alkuperäinen kuva.

Neljäs argumentti on yksinkertaisesti tekstuurin leveys ja viides argumentti on tekstuurin korkeus. Kuudentena argumenttina on border jonka täytyy aina olla nolla. Seitsemäntenä argumenttina annetaan funktiolle formaatti jossa sille annettava pikseli data on. Työssä käytetään BGRA-formaattia kun kuvat kopioidaan Kinectistä taulukkoon, joten sitä täytyy tässäkin käyttää. Kahdeksas argumentti kertoo funktiolle minkä tyyppistä pikseli data on, joten sille tulee spesifioida etumerkitön tavu, koska sitä käytetään Kinectistä saadussa datassa. Viimeisenä yhdeksäntenä argumenttina funktiolle annetaan osoitin joka osoittaa kuvan muistiosoitteeseen.

Lopuksi asetetaan vielä OpenGL:lle värit joita se käyttää väri puskurin puhdistamiseen. Tämä tehdään funktiolla glClearColor ja sille annetaan argumentteina järjestyksessä arvo väreille punainen, vihreä, sininen ja viimeisenä alpha-arvo. Alpha arvolla asetetaan värin läpinäkyvyys. Asetetaan myös glClearDepth-arvo jolla määritellään kuinka syvältä glClear-funktio tyhjentää puskurit kun se palauttaa ne oletusarvoihin. Viimeisenä ajetaan vielä glEnable jolla otetaan käyttöön GL_TEXTURE_2D eli sallitaan kaksiulotteisten tekstuurien luominen. (OpenGL reference 2016)

3.3.2 OpenGL-kameran alustaminen

Koska OpenGL on hyvin monipuolinen kirjasto käytettäväksi kuvien piirtämiseen ikkunaan, täytyy vielä määritellä mistä suunnasta tekstuuria katsotaan. Tätä varten määritellään kameralle asetukset kuvan neljätoista mukaisesti.

```
// Camera setup
glViewport(0, 0, width, height);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0, width, height, 0, 1, -1);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

KUVA 14. Kameran asetukset (Edward Zhang)

Ensimmäisenä asetuksena määritellään kameran ikkuna funktiolla glViewport. Tälle funktiolle annetaan argumenteiksi aluksi ikkunan vasemman alakulman koordinaatit X ja Y. Tämän jälkeen määritellään ikkunan korkeudeksi ja leveydeksi aikaisemmin luodun SDL-ikkunan korkeus ja leveys. Tämän jälkeen kerrotaan OpenGL:lle mitä matriisia tullaan käyttämään funktiolla glMatrixMode. Esimerkissä käytetään GL_PROJECTION-matriisi tilaa. Tämä käskee OpenGL:n piirtämään kuvan sellaisenaan ikkunaan ilman, että se yrittää tehdä kuvaan omaa syvyysvaikutelmaansa, koska kuva tulee jo kamerasta. Projection-matriisi siis muuttaa 3D kuvan 2D kuvaksi muuntamalla 3D kuvan pisteet 2D pisteiksi jotka voidaan piirtää näytölle. glLoadIdentity lataa nykyisen matriisin tilalle identity-matriisin. Identity-matriisi käytännössä palauttaa matriisin sen alkuperäiseen tilaan. Tämä on tarpeellista, koska OpenGL:n muut matriiseja käsittelevät funktiot vaikuttavat aina sen hetkisiin tiloihin. Identity matriisilla päästään takaisin alkuperäiseen matriisiin.

GlOrtho kertoo nykyisen matriisin ortografisella matriisilla. Käytännössä funktio saa kaikki tekstuurit näyttäytymään ikkunassa yhtä suurina riippumatta niiden etäisyydestä. Tätä käytetään siis yleisesti kaksi ulotteisten kuvien piirtämiseen OpenGL:n näkökenttään. Funktio ottaa argumentteina kuinka suuri alue on jolle tekstuuri piirretään ja skaalaa piirrettävät tekstuurit uudelleen, jotta ne näyttäisivät samalta vaikka ikkunan kokoa muutettaisiin. Ensimmäinen argumentti on vasen laita, toisena argumenttina annetaan alueen oikea laita. Kolmas argumentti on alueen pohja ja neljäs argumentti sen katto. Viidentenä argumenttina annetaan kuinka lähelle OpenGL-kamera voi nähdä minimissään ja viimeiseinä argumenttina kuinka kauaksi kyseinen kamera voi maksimissaan nähdä.

Lopuksi ladataan vielä GL_MODELVIEW-matriisi ja asetetaan se sen oletusarvoihin. Jos tarvitsisi näyttää jotakin muuta kuin kaksi ulotteista kuvaa, niin tätä matriisia käytettäisiin liittämään malli ympäröivään tekstuuriin.

3.3.3 OpenGL:n käyttäminen

Kun halutaan piirtää kuvia SDL-ikkunaan alku asetusten jälkeen, tulee aloittaa liittämällä tekstuurin ID-numero jälleen GL_TEXTURE_2D-elementtiin. Tämän jälkeen kutsutaan funktiota, jolla haetaan Kinectistä uusi värikuva ja tallennetaan se taulukkoon joka annetaan funktiolle argumenttina. Kun kuva on saatu Kinectistä ja se on tallennettu haluttuun taulukkoon, se voidaan siirtää kyseiseen tekstuuri elementtiin odottamaan piirtämistä. Tämän jälkeen ajetaan glClear-funktio, jolla tyhjennetään puskurit oletusarvoihin. Funktiolle annetaan argumenttina mitkä puskurit halutaan tyhjentää. Tyhjennykseen jälkeen aloitetaan piirtäminen komennolla glBegin. Funktiolle annetaan argumenttina GL_QUADS jolloin se tietää, että seuraavaksi sille annetaan koordinaatteja joista muodostetaan neliö jonka sisälle ladattu tekstuuri piirrettään.

```
void drawKinectData()
{
    glBindTexture(GL_TEXTURE_2D, textureId);
    GetKinectData(ColorImage);
    glTexSubImage2D(GL_TEXTURE_2D, 0, 0, 0, width, height, GL_BGRA_EXT, GL_UNSIGNED_BYTE, (GLvoid*)ColorImage);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glBegin(GL_QUADS);
    glTexCoord2f(0.0f, 0.0f);
    glVertex3f(0, 0, 0);
    glTexCoord2f(1.0f, 0.0f);
    glVertex3f(width/2, 0, 0);
    glTexCoord2f(1.0f, 1.0f);
    glVertex3f(width/2, height/2, 0.0f);
    glTexCoord2f(0.0f, 1.0f);
    glVertex3f(0, height/2, 0.0f);
    glEnd();
}
```

KUVA 15. OpenGL:llä kuvan piirtäminen ikkunaan (Edward Zhang)

glTexCoord2f-funktiolla määritellään neliö jonka sisälle tekstuuri piirretään. Funktiolle syötetään argumentteina kaikkien neljän kulman koordinaatit erikseen. glVertex3f-funktiolla määritellään mihin kohtaan ikkunaa kyseinen kuva piirretään. Kyseiselle funktiolle annetaan tarkat koordinaatit ikkunan sisältä. Tässä tapauksessa puolitetaan ikkunan leveys ja korkeus, jolloin saadaan piirrettyä tekstuuri ikkunan yläkulmaan ja jää tilaa vielä piirtää muita mahdollisesti tarvittavia debug kuvia. glEnd komennolla lopetetaan koordinaattien antaminen ja piirretään kuva osoitetun neliön sisälle.

4 KÄYTTÖKOHTEITA

Kinectille on jo olemassa monia käyttökohteita, mutta tässä on esiteltynä muutamia esimerkkejä mitä sillä on jo saatu aikaiseksi ja esitellään omia kehityskohteita.

4.1 NASA robotinohjaus

NASA on ottanut projektikseen luoda luonnollisempia tapoja ohjata robotteja avaruudessa. Yksi sovellus joka on ollut kehitteillä yhdistää virtuaalitodellisuuslasit Kinectin kanssa, jolloin voidaan ohjata robottikättä luonnollisemmin ohjaajan omaa kättä liikuttamalla. Tässä projektissa Kinectiä käytetään seuraamaan ohjaajan käden liikkeitä ja muutetaan kyseiset kädenliikkeet robottikäden liikkeiksi. Samalla virtuaalilaseilla näytetään käden liikkeet jolloin saadaan aikaan tunne siitä, että käytät omaa kättäsi kappaleiden käsittelyyn. (Nicole Lee 2016)

4.2 Traumaperäisen stressihäiriön tunnistaminen

Yhdysvaltojen DARPA(Defense Advanced Research Projects Agency) on käyttänyt Kinectiä sensorina projektissa, jossa sensoria käytetään seuraamaan sotilaita haastattelun aikana ja tulkitsemaan heidän kehonkieltään. Näiden ei tahdonalaisten liikkeiden perusteella voidaan tunnistaa stressihäiriön oireita vaikka sotilas kertoisi olevansa kunnossa. Kinectillä tarkkaillaan erityisesti päänliikkeitä, kädenliikkeitä ja henkilön asentoa sekä tapaa jolla henkilö puhuu haastattelussa. Projekti on toteutettu vanhemmalla Kinect versiolla ja tarkoituksena onkin ottaa käyttöön uudempi Kinect versio jossa on muun muassa paremmat kasvojen liikkeiden tunnistus ominaisuudet. (Kinect helps detect PTSD 2016)

4.3 Omia kehitysideoita

Jatkokehittelyn kannalta Kinectiä voisi esimerkiksi käyttää juurikin robotiikassa konenäkönä. Kinectin avulla luotaisiin ympäristöstä kolmiulotteista kuvaa ja sen perusteella ohjattaisiin robottia esteiden ohi. Kinectin rajoitusten takia robotissa tulisi kuitenkin olla lisänä sensoreita jotka näkisivät aivan robotin eteenkin, koska Kinectin sensori on epätarkka juurikin alle puolenmetrin päässä sensorista ja kameran katselukulma saattaa tuoda myös lisärajoituksia aivan maanrajassa oleviin esteisiin. Kinectin avulla voitaisiin myös reagoida käyttäjän esittämiin eleisiin tai mahdollisesti puhekomentoihin joiden avulla robottia voisi myös ohjata.

Toisena käyttökohteena voisi olla interaktiivisen näyttelyn luominen jossa Kinectiä käytetään havaitsemaan esiteltävän asian tai asioiden luokse saapuva ihminen ja tunnistamaan kyseisen ihmisen reagointi näyttelyn edessä. Kinectin avulla voitaisiin esimerkiksi havaita jos henkilö osoittaa jotakin tiettyä esinettä esittelytasolla ja tämän perusteella tuoda oheiselle näytölle lisä informaatiota osoituksen kohteesta. Samalla voitaisiin pitää tilastoa suosituimmista kohteista ja muokata näyttelyä saadun informaation perusteella. Samalla tekniikalla voitaisiin luoda esimerkiksi sovellus jonka avulla minkä tahansa pinnan voisi muuttaa kosketusnäytöksi.

5 POHDINTA

Työssä opeteltiin käyttämään Kinectiä omana yksikkönään, jotta sitä voitaisiin myöhemmin käyttää osana suurempaa kokonaisuutta tai jonkin toisen projektin osana. Aloitettiin käymällä läpi koodi esimerkit kuinka Kinectin eri ominaisuuksia käytetään ja kuinka saadaan sensoriin eloa sen jälkeen kun se on kytketty tietokoneeseen. Lopuksi tutustuttiin vielä monen data lähteen yhtäaikaiseen käyttöön. Työssä saavutettiin taso jossa voidaan todeta, että Kinect toimii ja siitä saadaan dataa ulos jonkin toisen ohjelman osan käytettäväksi.

Opeteltiin käyttämään VGB:tä ja Kinect-studiota jotta voitaisiin luoda omia elekirjastoja eleiden tunnistamisen helpottamiseksi. Kummankin ohjelman käyttämiseen on ohjeet Microsoftin toimesta, mutta niiden lisäksi tarvittiin hieman oma aloitteisuutta jotta saatiin ohjelmat toimimaan halutusti. Tätä prosessia yritettiin selventää ja siinä onnistuttiin mielestäni paremmin kuin Microsoftin omassa dokumentissa. Saatiin onnistuneesti myös luotua elekirjasto vaikeuksien jälkeen.

Kun kaikki Kinectin puolen ominaisuudet ja lisäohjelmat oli käyty läpi, keskityttiin datan näyttämiseen tietokoneen ruudulla. Tähän tarkoitukseen valjastettiin SDL- ja OpenGL-kirjastot. SDL on yksinkertainen kirjasto käyttää, eikä sen käyttämisessä esiintynytkään ongelmia. Sen sijaan OpenGL on hyvinkin monimutkainen ja vaativa kirjasto joka saatiin kuitenkin esimerkkien ja referenssin lukemisen jälkeen toimimaan niin, että Kinectistä saatu kuva ja data onnistuttiin piirtämään ruudulle.

Lopuksi esiteltiin vielä projekteja ja käyttökohteita mihin Kinectiä on jo käytetty ja mihin sitä voi tämän työn lukemisen jälkeen käyttää. Esiteltiin myös oma mielipide siitä mihin suuntaan projektia voitaisiin tästä jatkaa.

LÄHTEET

Kinect helps detect PTSD in combat soldiers. Artikkeli. Luettu 20.4.2016. https://blogs.msdn.microsoft.com/kinectforwindows/2015/07/01/kinect-helps-detect-ptsd-in-combat-soldiers/

Edward Zhang. Kinect v2 SDK C++. Käyttöohje. Luettu 1.3.2016. http://www.cs.princeton.edu/~edwardz/tutorials/kinect2/kinect0_sdl.html

Kinect for Windows Product Blog. Kuva. 18.4.2016. https://blogs.msdn.microsoft.com/kinectforwindows/2014/06/05/pre-order-your-kinect-for-windows-v2-sensor-starting-today/

Nicole Lee. NASA JPL controls robotic arm with kinect. Luettu 20.4.2016.

http://www.engadget.com/2013/12/23/nasa-jpl-control-robotic-arm-kinect-2/

OpenGL reference. Käyttöohje. Luettu 1.3.2016. https://www.opengl.org/sdk/docs/man/

SDL reference. Käyttöohje. Luettu 1.3.2016. https://wiki.libsdl.org/FrontPage?action=show&redirect=EtuSivu

Kinect Hardware. Luettelo. Luettu 1.3.2016. https://developer.microsoft.com/en-us/windows/kinect/hardware