Mikko Markkula

Kinect-sensori sovelluskehityksessä

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Tietotekniikan koulutusohjelma
Insinöörityö
26.10.2011

Metropolia

Tiivistelmä

Tekijä Otsikko Sivumäärä Aika	Mikko Markkula Kinect-sensori sovelluskehityksessä 47 sivua 26.10.2011
Tutkinto	Insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	ohjelmistotekniikka
Ohjaaja	yliopettaja Jarkko Vuori

Kinect on vuonna 2010 julkaistu Xbox360-pelikonsolin lisäohjain. Se on Microsoftin vastaus Nintendo Wiin aloittamaan liikkeentunnistusbuumiin, mikä sai konsolipelaajat ylös sohvalta ja liikkeelle. Ohjaimen toiminta perustuu koko vartalon seurantaan ilman fyysistä ohjainta ja puheohjaukseen. Laitteesta tuli yllättävä myyntimenestys, ja se nousi Guinnessin ennätysten kirjaan nopeimmin myyneenä elektroniikkalaitteena. Pian julkaisun jälkeen harrastelijat kehittivät avoimen lähdekoodin ajurit Kinectille, mahdollistaen laitteen käytön PC:llä. Vastauksena harrastelijoiden innokkuuteen Microsoft julkaisi viralliset kehitystyökalut.

Tämän työn tarkoitus oli tutkia Kinectin ominaisuuksia konsolin ohjaimena ja käydä läpi, minkälaisia ohjelmointialustoja ja -mahdollisuuksia on tarjolla. Lopulta kehitettiin sovellus, joka hyödyntää kaikkia Kinectin ominaisuuksia eli liike- ja äänitunnistusta. Sovellusta luodessa selvitettiin, mitä haasteita luonnollista käyttöliittymää kehittäessä tulee vastaan.

Esimerkkisovelluksena toimi yksinkertainen tietovisa, jossa ruudulta voi valita aiheita ja vastauksia painamalla tai puhumalla ääneen haluttu komento tai vastaus. Se toteutettiin virallisilla kehitystyökaluilla, koska niillä pääsi välittömästi käsiksi tietoon koko vartalon liikkeestä.

Esimerkkisovellus täytti ja ylitti sille asetetut vaatimukset. Kehitystyökalujen kanssa oli erittäin nopea kehittää luonnollinen käyttöliittymä, jonka käyttö oli intuitiivista ja yksinkertaista. Kinect itse toimi testikäytössä hyvin, lukuun ottamatta ongelmia auringonvalon ja yliherkän puheentunnistuksen kanssa. Laite onkin mainio leikkikalu harrastelijoille, mutta varsinaista läpilyöntiä ei vielä näillä ominaisuuksilla ole odotettavissa konsolimarkkinoiden ulkopuolella.

Avainsanat	Kinect, Xbox 360, liikkeentunnistus

Metropolia

Abstract

Author Title Number of Pages Date	Mikko Markkula Kinect sensor in software engineering 47 pages 26 October 2010
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor	Jarkko Vuori, Principal Lecturer

Kinect is a motion controller released for the Xbox 360 game console. It is Microsoft's response to the dominance of Nintendo Wii on the motion control market. The device is based on full-body motion tracking and speech recognition. There is no physical controller at all. Surprisingly, Kinect became a bestseller, and it claimed the Guinness World Record being the fastest selling consumer electronics device. Shortly after the release, enthusiasts managed to reverse engineer open-source drivers for Kinect, enabling it on various platforms. Microsoft released official development tools in response.

The purpose of this thesis was to study the properties of the Kinect controller as a gaming peripheral and go through different programming platforms made available by official and unofficial drivers. The final goal was to create an application that would use all the Kinect functions, namely full-body motion tracking and speech recognition. In process, another goal was to study what it takes to create a natural user interface.

The application functioning as the prototype was a simple quiz in which the player could select topics and responses by clicking them or saying them aloud. The application was implemented with the official development tools, because it allows immediate access to the skeletal tracking.

The application met and exceeded all the requirements set for it. Creating the natural user interface with the development tools was fast and easy. Kinect worked well in most of the test scenarios but had problems with sunlight, distance demanded by the sensor technology and oversensitive speech recognition. In conclusion, the device is a wonderful toy for amateurs and enthusiasts although it has to be admitted that most likely Kinect will not penetrate the PC market in its current stage.

Keywords	Kinect, Xbox 360, full-body motion tracking

Sisällys

1 Johdanto

Pelikonsolien siirtyessä seitsemänteen sukupolveen yksi kolmesta kilpailevasta yrityksestä otti valtaisan riskin ja valitsi uuden konsolinsa ohjaimeksi laitteen joka toimii osoitusperiaatteella ja tunnistaa liikkeen kolmessa ulottuvuudessa. Nintendo esitteli Wii:ksi nimetyn konsolinsa vuoden 2005 E3-messuilla ja uuden ohjaimen myöhemmin samana vuonna Tokion pelimessuilla. Esitelty uusi konsoli herätti hilpeyttä varsinkin nimensä puolesta, mutta uusi ohjaustyyli oli kuitenkin ennennäkemätöntä ja jo seuraavan vuoden E3-messuilla laitetta pidettiin pitkälti menestyksenä. [1.]

Wii:stä tuli ilmiö sekä nykyisen konsolisukupolven ylivoimainen voittaja. Myyntitilastoissa ero seuraavaksi parhaiten myyneeseen konsoliin on jopa 30 miljoonaa yksikköä [2]. Syy menestykseen oli uusi ohjaamistyyli, joka oli intuitiivinen ja helposti lähestyttävä, joten pelaamisen pariin löysi täysin uusi ryhmä, ns. kasuaalipelaajat. Ohjaustapa pakoitti myös pelaajat ylös sohvalta, joten yleinen suhtautuminen pelaamiseen parani. Kilpailevat konsolit eli Microsoftin Xbox360 ja Sonyn Playstation 3 keskittyivät konsolikehityksessä paljon perinteisempään malliin, missä koneen suoritustehon ja lisäominaisuuksien kasvattaminen oli suurin innovaatio. Molemmat laitteet pystyvät teräväpiirtografiikkaan, toisin kuin teknisesti heikompi Nintendon Wii. Tämä nosti valmistuskustannuksia, ja varsinkin Sony joutui myymään Playstation 3 -konsolia tappiollisesti, toivoen että pelimyynti korvaa tappiot. Nintendolla ei tätä ongelmaa ollut eikä heikompi graafinen taso heikentänyt myyntiä. [3.]

Oli ilmiselvää, että liikkeentunnistukseen perustuva pelaaminen oli tullut jäädäkseen, joten Sony ja Microsoft lähtivät kehittämään omia haastajiaan Wiille. Tähän meni kuitenkin neljä vuotta, ja Nintendo oli jo karannut myynnissä molemmilta yrityksiltä. Siinä missä Sony rakensi Wiin ohjausmetodia muistuttavan ohjaimen, Microsoft otti suuremman riskin omalla ohjaimellaan, missä ei ole fyysistä pideltävää kappaletta ensinkään, vaan liikkeentunnistus pohjautuu seurattavaan luurankomalliin aina kahdelle pelaajalle asti. Pelikansa otti uutiset tästä ohjaimesta vähäisellä innostuksella vastaan koska fyysistä pideltävää ohjainta pidetään ehdottoman tärkeänä tarkalle pelaamiselle. Kinectistä tulikin yllättävä menestys, koska erikoinen pelien ohjaustapa ja futuristiselta tuntuva käyttöliittymän hallinta äänen ja liikkeen avulla vetosi ihmisiin. [4.]

Koska laite kiinnitetään Xbox360-konsoliin USB-kaapelilla, kekseliäät mielet tajusivat, että laitteen liikenne on mahdollista purkaa ja selvittää. Pian laitteen julkaisun jälkeen pidettiin kilpailu siitä, kuka ensimmäisenä onnistuisi kehittämään avoimen lähdekoodin ajurit Kinectille. Pari viikkoa julkaisun jälkeen ilmaantui ensimmäiset toimivat, tosin alkeelliset laiteajurit, joilla pääsi käsiksi syvyyskameraan. Microsoft ei kuitenkaan haastanut ketään oikeuteen, vaan tajusi edun siitä, että laitteelle löytyi uusi innokas käyttäjäryhmä. Yrityksen edustajien mukaan laitteen arkkitehtuuri oli tahallaan jätetty avoimeksi juuri tämänlaisten mahdollisuuksien takia. Tarkemman tulkinnan mukaan hakkeroinnista ei ollut kyse, koska mihinkään laitteen käyttämiin tunnistusalgoritmeihin ei päästy käsiksi eikä laitetta itsessään muokattu mitenkään. [5.]

Harrastelijayleisön innostus oli valtava, joten Microsoft päätti julkaista viralliset kehitystyökalut Kinectille. Kehitysasteella olevat työkalut julkaistiin kesäkuussa 2011, eikä niille toistaiseksi ole olemassa kaupallista lisenssiä. Microsoft on myös lupaillut PC-julkaisua Kinectistä, koska se on nykyiseltään optimoitu pelikonsolin ohjaimeksi eikä pöytäkoneen liitännäiseksi. [6.]

Tämän insinöörityön tarkoitus on esitellä Kinectin ominaisuuksia pelikonsolin ohjaimena ja tutkia, minkälaista ohjelmointimahdollisuuksia ja ympäristöjä eri alustoille löytyy. Tavoitteena on myös kehitellä lopulta esimerkkiohjelma, jossa hyödynnetään kaikkia ohjaimen ominaisuuksia. Työssä kartoitetaan, onko tämä teknologia ylipäätänsä toimivaa ja käyttökelpoista käytännönläheisestä näkökulmasta.

2 Kinectin esittely

Microsoft toi uuden ohjaintekniikkansa julkisuuteen ensimmäistä kertaa vuoden 2009 E3-messuilla Los Angelesissa. Ohjain kulki silloin vielä nimellä Project Natal ja sen oli tarkoitus haastaa liikkeentunnistuksen yksinvaltias Nintendo sekä tuoda lisää elinikää jo vuonna 2005 julkaistulle Xbox 360-konsolille. Messuilla esiteltiin erilaisia ominaisuuksia, kuten koko vartalon seurantaa, kasvojen tunnistamista, videokeskusteluita sekä reaalimaailman kappaleiden mallintamista. Laitteella pelaaminen tapahtuisi seuraamalla koko vartalon liikettä aina kahteen pelaajaan asti. Seuraavan vuoden messuilla nimi muuntui Kinectiksi ja julkaisupäivä annettiin loppuvuodelle.

Konsoliyleisön ja varsinkin paljon pelaavan ydinjoukon reaktio Kinectiin oli erittäin negatiivinen, koska fyysistä ohjainta ja fyysisiä nappuloita pidetään välttämättömänä tarkalle ohjaukselle. Laitteesta nousi esille myös erilaisia huolenaiheita, kuten se että pelaaminen onnistuisi ainoastaan seisaaltaan, eli Kinect ei tunnista istuvia hahmoja. Esitellyissä peleissä oli myös ilmeistä, että laitteen vasteaika ei soveltunut kilpailuhenkiseen pelaamiseen. Microsoftin tarkoitus oli kuitenkin laajentaa käyttäjäryhmäänsä ja vedota samaan pelaajaryhmään, jotka ihastuivat Wii-konsoliin eivätkä löytäneet Xbox 360:sta itselleen mitään tarjontaa. Kinectin suunnittelussa panostettiin myös tyylikyytteen, kuten kuvasta 1 käy ilmi.

Kuva 1. Kinect-liikkeentunnistusohjain

Julkaisu tapahtui joulusesonkina 2010 ja lanseerauspelejä luvattiin 15 kappaletta, joista kaikki eivät ilmaantuneet myyntiin asti. Uudet ominaisuudet sai käyttöjärjestelmään

190 megatavun päivityksellä. Laitteelta puuttui oma hittipelinsä, mutta se oli kuitenkin jotakin uutta ja hämmästyttävää suurimmalle osalle yleisöstä. Kriitikoiden mielipiteet jakautuivat mutta yleisesti ottaen vastaanotto oli positiivista. Lähtöhintana oli 150 dollaria, mitä pidettiin melkoisen korkeana, koska jo pelkän konsolin sai yksin 200 dollarin hintaan. Suhteutettuna siihen, että Kinect on käytännössä kahden pelaajan ohjain monilla lisäominaisuuksilla, oli se kuitenkin aika edullinen. Hinnasta huolimatta se meni myynnissä Guinnessin ennätysten kirjaan nopeimmin myyneenä elektroniikkalaitteena, ja viimeisten julkaistujen tilastojen mukaan sitä on myyty yli 10 miljoonaa kappaletta. [4.]

2.1 Xbox 360:n ominaisuudet

Kinect ei toimi pelkästään peliohjaimena vaan se on myös oleellinen osa Microsoftin ns. 360-kokemusta. Tämä tarkoittaa yksinkertaistettuna sitä, että konsolin on tarkoitus olla kodin keskeinen mediakeskus, josta löytyy jotakin kaikenikäisille käyttäjille kaikenlaisiin käyttötarpeisiin. Elinkaarensa aikana konsolin Dashboard-käyttöliittymä on käynyt läpi monta iteraatiota, joista viimeisin lisäsi Kinect-ominaisuudet. Myös konsolista on julkaistu uusi pienempi malli, joka korjasi vanhan mallin ongelmia, pahimpana lämmöntuotto, joka rikkoi laitteita sekä kova metelöinti varsinkin laitteen lukiessa levyä optisesta asemasta. Uudessa mallissa on myös oma USB-porttinsa Kinectille, siinä missä vanhan konsolin kanssa ohjain tarvitsee lisävirtaa ulkopuolelta. [4.]

2.1.1 Käyttöliittymä

Yksi esitellyistä ominaisuuksista oli uusi Kinectillä ohjattava käyttöliittymä, joka toimii täysin ilman fyysistä ohjainta. Visuaalisesti muutos ei ollut iso aikaisempaan versioon ja Kinectille on oma pääikkunansa, mistä navigoidaan eteenpäin. Tavallinen vanha käyttöliittymä on yhä tarjolla, eikä se tue kaikkia Kinectin toimintoja.

Kuva 2. Kinect Hub [7]

Käyttäjä voi siirtyä kuvassa 2 esiintyvään Kinectin pääikkunaan heiluttamalla kättä, sanomalla Xbox Kinect tai navigoimalla perinteisesti peliohjaimella tavallisesta käyttöliittymästä. Ruudulta voi valita ikoneja pitämällä kättä ruudun päällä tai sanomalla ääneen sen mitä näkee. Äänikomennoilla voi mm. pysäyttää videon toiston tai avata DVD-aseman mutta ne tarvitsevat aina etuliitteen "Xbox" millä eliminoidaan mahdollisia virhetilanteita vääristä tunnistuksista.

2.1.2 Pelit

Liiketunnistusbuumin taustalla oli into siitä, että pelejä ei enää pelata staattisesti kyyhöttäen sohvalle, vaan aktiivisesti liikkuen. Tästä syystä monet Kinectin julkaisupeleistä olivat paljon liikettä vaativia pelejä, kuten tanssipelejä ja kuntoilupelejä. Malli näihin peleihin kopioitiin selkeästi Nintendon menestyspeleistä, kuten Wii Sportsista, mikä on kokoelma erilaisia pieniä urheilupelejä. Yksikään peli ei saanut täysin yksimielistä ylistystä ja Kinectiltä puuttui oma kriitikoiden yksimielisesti ylistämä hittituote. Yhdessäkään pelissä ei ollut omaa tarinallista rakennetta vaan kaikki perustuivat varsin rajalliseen pelitilanteeseen.

Kuva 3. Kinect Adventures [8]

Tilastollisesti eniten myynyt peli on Kinect Adventures (kuva 3), joka tulee jokaisen myydyn sensorin mukana. Se on viiden urheilu- ja seikkailupelin kokoelma, eli se muistuttaa hyvinkin paljon Nintendo Wiin peruspelejä. Arvosteluissa se ei menestynyt mutta se on erinomainen näyte siitä mihin, kaikkeen Kinect pystyy. [2.]

Kuva 4. Dance Central [9]

Peliarvosteluissa parhaiten menestynyt Kinect-peli on Guitar Herosta ja Rock Bandista tutun Harmonixin tanssipeli Dance Central (kuva 4). Pelissä tarkoituksena on seurata ruudulla liikkuvia tanssiliikkeitä, jotka pitää toistaa mahdollisimman tarkasti. Suosiota ei ole vaikea ymmärtää, koska tanssipeli on ihanteellinen sovellus koko vartalon seuran-

nalle. Tanssi- ja musiikkipelit eivät ole mikään uusi ilmiö vaan ovat lyöneet itsensä läpi jo kauan aikaa sitten. Nykyisistä tanssipeleistä suurin kilpailija on Wii:lle kehitetty Just Dance, mikä myy hurjia määriä heikosta laadustaan huolimatta. Dance Centralin myynti on yltänyt miljooniin kappaleisiin, ja peli sai hiljattain jatko-osan, mistä povataan Kinect-hittituotetta alkavalle lomakaudelle. [10.]

2.1.3 Tulevaisuus

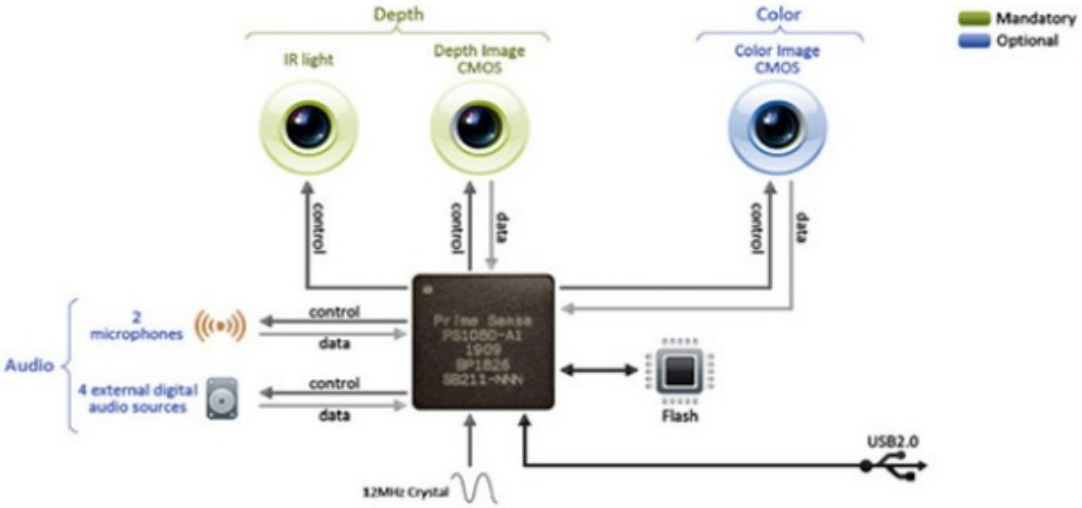
Kinectille lupaillaan firmware-päivitystä, jonka tulisi lisätä sormenpäiden liikkeen seuraaminen ohjelmistoon. Ongelmana on kuitenkin syvyyskuvan pieni resoluutio, jota pakataan konsolille lähettäessä rajusti. Siitä ei ole mitenkään mahdollista kaivaa esiin tietoa yksittäisten sormien liikkeestä tai ranteen kääntymisestä. Ratkaisuna Microsoft etsii tapoja lisätä Kinectiltä laitteelle kulkevan tiedon määrää eri pakkausteknologioilla. Firmware säätelee myös syvyyssensoria, ja tällä hetkellä se on rajoitettu 30 ruutuun sekunnissa 320 x 240 pikselin resoluutiolla. Piiri, mihin Kinect perustuu, kykenisi kuitenkin käsittelemään huomattavasti suuremman määrän tietoa.

Mikäli resoluution nostaisi sensorin maksimiin eli 640 x 480 pikseliin, datan määrä nelinkertaistuisi ja Kinect voisi havainnoida liikettä tarkemmin. Nelinkertainen ruutu tarkoittaa kuitenkin huomattavasti suurempaa työmäärää, ja suurin osa pelikonsolin tehoista täytyy kuulua itse peleille. Microsoft ei ole julkaissut tarkkaa lukua Kinectin toimintojen vaatimasta tehomäärästä mutta on viitannut, että vaadittu prosessointiteho on saatu puristettua alle kymmeneen prosenttiin. Tarkkaa päivämäärää tälle päivitykselle ei ole kuitenkaan julkaistu. [11; 12.]

Yleinen huolenaihe laitteesta on ollut se, että Kinect mielletään jatkossa pelkästään Dance Central -koneeksi, koska se on eräs yleisimmistä syistä laitteen hankkimiseen. Tämän vuoden E3 -messuilla Microsoftin konferenssin pääteema oli Kinect ja sen tuki pelaajien ydinjoukolle. Monesta isomman budjetin pelistä löytyy jatkossa jonkinlaisia Kinect-ominaisuuksia, mutta laite ei ole yhdellekään pelille pakollinen. Vain muutama isomman budjetin peli, joka käyttää konsolin täyttä graafista potentiaalia ja on täysin riippuvainen Kinectistä, esiteltiin. [13.]

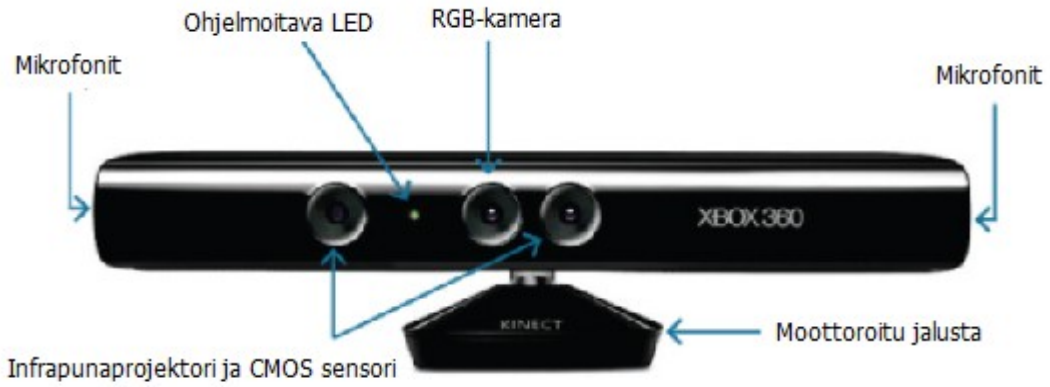
2.2 Tekniset ominaisuudet

Kinect pohjautuu israelilaisen PrimeSensen teknologiaan. Se kykenee käsittelemään syvyyskuvaa, värikuvaa ja ääntä yhden piirin kautta. Syvyyskuva on matriisi, jossa alkion arvo kertoo sensorin tunnistaman etäisyyden kyseiselle pisteelle. Etäisyyden laskemisesta kerrotaan kappaleessa 2.3. Pelaajien havainnointi syvyyskuvasta tapahtuu myös piirillä, ja tieto tästä viedään Microsoftin kirjastojen käsiteltäväksi, jotka sijaitsevat päätelaitteessa eli Xbox 360 -konsolilla tai PC:llä. Luurankomallien muodostaminen tapahtuu näissä kirjastoissa, kuten kaikki muukin tarkempi tunnistaminen, esimerkiksi puhe. PrimeSensen väitteiden mukaan sensori kykenisi tunnistamaan 1 cm:n kappaleita kahden metrin päästä, johon käytännössä ainakaan Kinectin kanssa tähän ei olla vielä päästy. [4; 14.]



Kuva 5. PrimeSensen PS1080-toimintakaavio [15]

Kinectin ytimessä on PrimeSensen PS1080 SoC (System on a chip), joka tarjoaa synkronisoitua syvyyskuvaa, värikuvan sekä äänivirran, kuten kuvan 5 toimintakaavioista näkyy. PS1080 yhdistetään USB 2.0 -liitännällä isäntälaitteeseen, eikä vastaanottavan laitteen tehoista tehdä minkäänlaisia oletuksia. Kaikki syvyyden saamiseen käytettävä algoritmit pyörivät PS1080:lla ja isäntälaitteessa pyörii ainoastaan USB-liikenteeseen tarvittavat ominaisuudet. Tämä takaa, että syvyyden tunnistamista voi käyttää rajallisten isäntälaitteiden kanssa, mikä ei tässä tapauksessa ole kyseessä. Kinect on toistaiseksi ainut laite, joka käyttää tätä piiriä. Kinect toimii firmwarella, joten ominaisuuksia voi lisätä tai muokata milloin tahansa, tietenkin laitteiston teknisten rajoitteiden puitteissa. [16.]



Kuva 6. Kinectin ominaisuudet [4]

Infrapunaprojektori ja CMOS-sensori

Pelialueen syvyyden tunnistaminen perustuu infrapunaprojektorin sekä CMOS-kennon yhteistoimintaan. Näillä voidaan tallentaa kolmiulotteista mustavalkoista kuvaa missä tahansa valaistusympäristössä. Tunnistusetäisyys on myös säädettävissä pelaajan sijaintiin automaattisesti, ja Kinectin ohjelmisto osaa kalibroida pelialueen fyysisen ympäristön mukaan, mikäli sieltä löytyy esteitä kuten huonekaluja. Resoluution syvyyskameralle sanelee firmware, ja toistaiseksi konsolikäytössä se on rajattu 320 x 240 pikseliin 30 ruutua sekunnissa. Se ei kuitenkaan ole tekninen maksimi vaan laite voi käyttää myös 640 x 480 pikselin resoluutiota. [4; 17; 18.]

RGB-videokamera

Kinectissä on myös perinteinen RGB-kamera, jonka resoluutio on 640 x 480 pikseliä ja ruudunpäivitysnopeus 30 sekuntia. Sitä voidaan käyttää pelitilanteen nauhoittamiseen, videokeskusteluihin, ja se on olennainen osa kasvojen tunnistamisessa.

Mikrofonit

Kinectissä on neljä mikrofonia, joilla on mahdollista kaapata korkealaatuista ääntä, poistaa siitä akustinen kaiku ja paikallistaa äänilähde. Se on tärkeä ominaisuus tilanteessa, jossa pelaajia on useita ympäri huonetta ja taustamelua on paljon. Neljä mikro-

fonia sijaitsevat Kinectin alaosassa vierekkäin. Kun ääni saapuu jokaiseen mikrofoniin eri aikaan, siitä voidaan päätellä mistä ääni tuli, ja suunnata mikrofoni äänilähdettä kohti. Näin saadaan mahdollisimman hyvälaatuista äänikaappausta. Puheen tunnistaminen ei itsessään ole mikään Kinectin ehdoton ominaisuus, vaan se voitaisiin toteuttaa millä tahansa mikrofonilla. Kinectin teknologiat mahdollistavat kuitenkin äänen kaappaamisen olosuhteissa, johon tavallinen mikrofoni ei pysty. Jokainen mikrofoni äänittää 16-bittistä ääntä 16 kHz:n näytteenottotaajuudella. [19.]

Moottoroitu jalusta

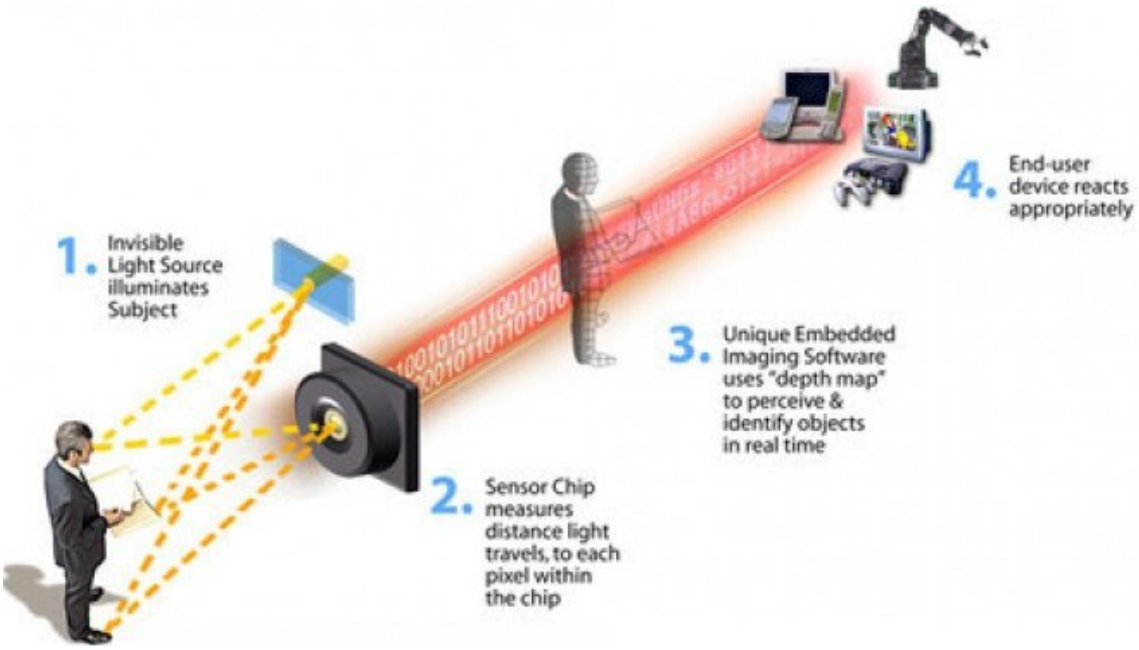
Kinectin jalusta on moottoroitu, ja se voi kallistaa sensoria 28 astetta ylös- tai alaspäin. Kalibroitaessa ohjainta laite etsii itselleen optimaalisen katselukulman riippuen siitä, mihin paikkaan se on asennettu olohuoneessa.

2.3 Syvyyskameran toiminta

Infrapunaprojektori valaisee huoneen pistekuviolla, joka noudattaa tarkkaa kuviota. Kuvio on tärkeä, koska sen mukaan päätellään pisteen sijainti. Ihmissilmä ei havaitse infrapunavaloa, joten pisteiden havaitsemiseen tarvitaan erityislaitteita, kuten pimeäkiikaria kuvassa 24. [18.]

Kuva 7. IR-pisteet nähtynä pimeäkiikarilla [20]

Kuvion pohjalta voidaan päätellä, miten kauan valolta kesti liikkua kohteeseen ja takaisin CMOS-kennoon. Kenno muuntaa valon energian jännitteeksi, josta voidaan erilaisilla algoritmeilla muodostaa syvyyskuva. Kuva 8 esittää toiminnan yksinkertaistettuna.



Kuva 8. Kinectin toiminta [21]

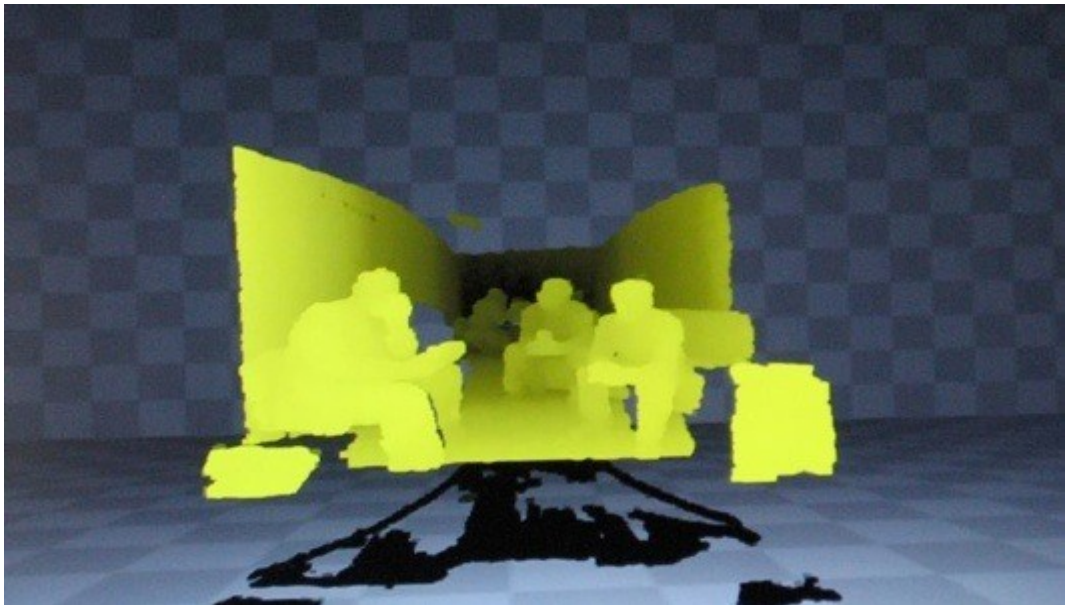
Syvyyskuva kulkee isäntälaitteelle pelkkänä tietona siitä, missä piste sijaitsi ja miten kaukana se oli. Se voi kuitenkin kertoa, kuuluuko kyseinen pikseli hahmolle, joka tunnistettiin pelaajaksi. Tämä tunnistusteknologia takaa sen, että on yhdentekevää, onko kyseessä pimeä vai valaistu huone. [21.]

2.4 Rajoitteet ja kritiikki

Kinect ei selvinnyt täysin puhtain paperein julkaisuhetkellä, vaan laitteen käytössä on ilmennyt erilaisia puutteita tai ongelmia. Suurinta närää on aiheuttanut tunnistus, joka ei salli istumista pelatessa, vaan vaatii seisoma-asentoa. Se toimii kuitenkin monessa tapauksessa myös istualtaan. Laitteella ei ole mitään rajoitteita, ja ääni- sekä kuvavirtoihin pääsee suoraan käsiksi ja niitä voi käyttää haluamallaan tavalla. Pelkkä käsiohjaus olisi siis mahdollista, jos vain kirjastotuki saadaan.

2.4.1 Tekniset rajat

Kinectin pelialue on noin kuuden neliömetrin kokoinen, ja pelatessa etäisyyden tulisi olla 1,2 – 3,5 metriä. Sensori pystyy tunnistamaan etäisyyksiä kauempaa, aina kuuteen metriin asti. Katselukulma on 43 astetta pystysuunnassa ja 57 astetta vaakasuunnassa. Mikäli kappaleen vie liian lähelle sensoria, syvyyskuvassa ilmenee erikoinen ilmiö ja kappale näkyy kahdesti, ikään kuin varjona. Lähimmäksi etäisyydeksi on mainittu 0,7 metriä, mistä sensori pystyy tunnistamaan kappaleen virheettä. Latenssi on myös ongelma pelatessa ja yleensä se on noin 200 – 300ms:n luokkaa, mikä on jo liikaa kilpailuhenkiseen pelaamiseen. Pelialueella ei tulisi olla heijastavia pintoja, koska ne aiheuttavat ongelmia syvyyskuvan kanssa. Mikäli syvyydestä ei saada tarkkaa tietoa, kyseinen pikseli saa arvon, joka viittaa aina etäisyyteen, mitä ei tiedetä. Kuvassa 9 on nähtävissä tämänlainen alue takavasemmalla. [22.]



Kuva 9. Kinectin havainnoima syvyyskuva 3D-mallina [14]

On myös hyvä muistaa, että laite ei voi mitenkään nähdä kappaleiden taakse. Kuvasta 9 saa hyvän käsityksen siitä, mitä syvyyskamera näkee. Luurankotunnistus on mahdollista tehdä kahdelle pelaajalle, vaikka itse sensori voi havainnoida useampia hahmoja ruudulla. Näiden hahmojen käsittely vaatisi kuitenkin valtavan määrän laskentatehoa. [14; 17.]

2.4.2 Ulkopuoliset häiriötekijät

Pelialueen syvyyden tunnistaminen perustuu infrapunavalon mittaamiseen, joten auringonvalo aiheuttaa häiriötekijöitä jäljittämiseen, koska auringon infrapunasäteily on samalla aallonpituudella Kinectin IR-projektorin kanssa. Onkin suotavaa, että sensori tai pelaaja ei ole suorassa auringonvalossa. Laitteen luurankotunnistus toimii pimeässä, mutta kasvojen tunnistaminen vaatii valaistusta. Tummaihoisilla pelaajilla onkin ollut ongelmia kasvontunnustusominaisuuden kanssa. [23.]

Äänitunnistus pyrkii eliminoimaan suurimman osan mahdollisista häiriötekijöistä. Teknologia toimii, ja ääni tunnistetaan taustamelunkin läpi. Pelatessa valittu asustus vaikuttaa ratkaisevasti siihen, miten hyvin tunnistus toimii. Aamutakilla ja verkkarihousuilla on selkeä ero.

2.4.3 Kinect vs EyeToy

Sony julkaisi kuudennen sukupolven Playstation 2 -pelikonsolille lisäohjaimen nimeltä EyeToy, joka on melko yksinkertainen web-kamera, johon on yhdistetty mikrofoni. Sama laite julkaistiin seitsemännen sukupolven Playstation 3 -konsolille nimellä Playstation Eye. Toiminnallisesti se on lähes tulkoon sama laite pienillä parannuksilla.

Useassa keskustelussa tai artikkelissa törmää väitteeseen, että Kinect ei ole muuta kuin hienostuneempi EyeToy. Väite ei ole ollenkaan perätön, sillä molempien laitteiden antama kokemus on hyvin samanlainen. Playstationille löytyi jopa pelejä, jotka kykenivät koko vartalon tunnistamiseen ja ääniohjaukseen.

Teknisesti ero on kuitenkin huomattava. EyeToy ei kykene havainnoimaan pelialueen syvyyttä mitenkään. Koska EyeToy tunnistaa liikettä tavallisesta värikamerasta, se vaatii hyvät valaistusolosuhteet eikä näe pimeässä huoneessa mitään. Laitteiden isoin ero on kuitenkin Microsoftin Kinectille antama kirjastotuki, jossa on kaikki tarvittavat ominaisuudet, kuten ääni- ja luurankotunnistus. [24; 25.]

3 Kinectin hakkerointi

Koska laite yhdistetään konsoliin USB-piuhalla, kekseliäät alan ammattilaiset tajusivat, että USB:ta tutkimalla ohjaimen saisi toimimaan millä tahansa halutulla alustalla. Kaiken lisäksi rakenne oli jätetty avoimeksi eikä liikenteessä ollut suojauksia. Ei mennyt pitkään, että nämä ajurit ilmaantuivat, ja ne ovat olleet jatkuvan kehityksen alla siitä lähtien. Kehitystä nopeutti se, että ensimmäisistä jotenkin toimivista avoimen lähdekoodin ajureista luvattiin muutaman tuhannen dollarin palkkio. Tarkemmin USB-liikenteen purkamiseen tässä insinöörityössä ei tutustuta, vaan keskitytään seuraamuksiin. [26.]

Virallisen Microsoftin lausunnon mukaan laitetta ei varsinaisesti hakkeroitu, koska se tarkoittaisi sitä, että joku pääsisi laitteen sisällä piileviin algoritmeihin käsiksi ja onnistuisi käyttämään niitä. Näin ei kuitenkaan käynyt, vaan joku onnistui luomaan avoimen lähdekoodin ajurin PC:lle. Ajuri avaa käytännössä vain USB-yhteyden ja lukee sensorin raakapalautetta Kinectistä käsittelemättä sitä mitenkään. Kinectin arkkitehtuuri oli tahallaan jätetty avoimeksi juuri tämänkaltaisia mahdollisuuksia silmällä pitäen. Sen sijaan, että Microsoft haastoi "hakkeroijat" oikeuteen, antoi yritys tukensa Kinectin PC-kehittäjille, koska harrastelijapiirien innostus ja potentiaali oli ilmeistä. Luvatut viralliset kehitystyökalut ilmaantuivat kesäkuussa mutta ne eivät vielä tarjonneet kaupallista lisenssiä, joka on luvassa joskus tulevaisuudessa todennäköisesti kehitystyökalujen lopullisen version kanssa. [27.]

3.1 Harrastelijaprojekteja

Harrastelijat sekä ammattilaiset ovat kehittäneet mitä mielikuvituksellisimpia ohjelmia hyödyntäen Kinectiä. Näitä löytyy iso määrä sivustolta Kinecthacks.net, jonka tarjonnasta osa on amatöörien tuherruksia mutta osa on nerokkaita viritelmiä. Ne ruokkivat mielikuvitusta siitä, mitä tulevaisuuden käyttöliittymä saattaa olla tai mihin kaikkeen Kinect oikeasti taipuu. Niitä katsellessa ihmettelee helposti, miksi Microsoft ei ole toteuttanut joitakin ominaisuuksia oman konsolinsa käyttöliittymässä. On kuitenkin eri asia tehdä harrastelijaprojektia kuin käyttöliittymää, joka päätyy miljooniin koteihin ja jonka on toimittava olosuhteista riippumatta. [28.]

3.1.1 Kinect Minority Report

Kinect ei ole rajoitettu pelkästään luurankomallin tunnistamiseen. Koska se toimii pääsijaisesti peliohjaimena, käsien pikkutarkkaa tunnistamista ei toistaiseksi ole lisätty pelikonsolin ominaisuuksiin. Tämä ei kuitenkaan tarkoita, etteikö se olisi mahdollista, kuten kuvan 10 MIT:ssä tehty esimerkki osoittaa.

Kuva 10. Käyttöliittymä, jota voi hallita sormen liikettä seuraamalla [29]

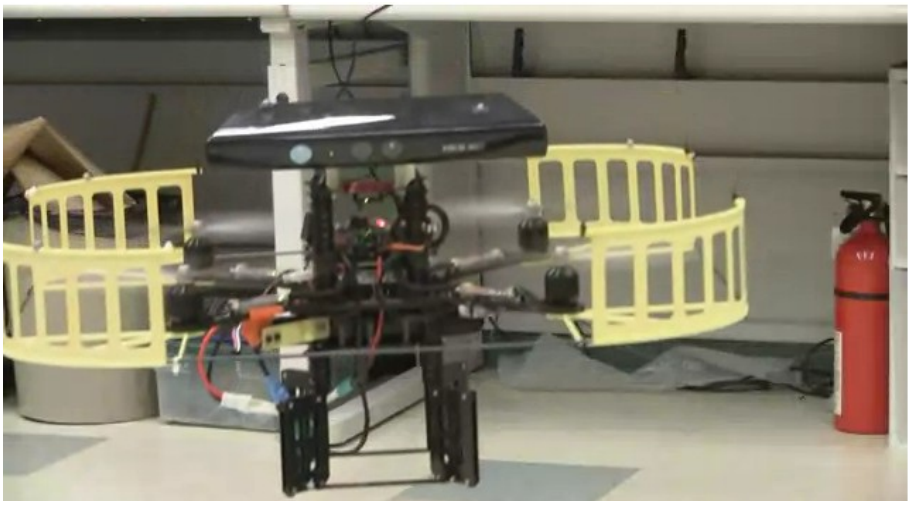
Inspiraatio ja nimi tälle käyttöliittymälle on tullut elokuvasta Minority Report, jossa käyttöliittymiä hallittiin intuitiivisilla käsieleillä. Esimerkissä melko yksinkertaista kuvagalleriaa hallitaan seuraamalla kahden käden liikkeitä.

Ruudulta voi valita kuvia käsiteltäväksi ja niitä voi venyttää, pyörittää, suurentaa ja pienentää sormieleitä seuraamalla. Ohjelma seuraa käyttäjän molempia käsiä ja ruudulle voi vetää useita kuvia käsiteltäväksi samanaikaisesti.

Se tunnistaa käsien liikkeet 60.000 pisteen verkosta 30 kertaa sekunnissa ja mahdollistaa luonnollisen, reaaliaikaisen käyttökokemuksen. [29.]

3.1.2 Lentävä Kinect

Syvyyskuvaa voi myös käyttää ympäristön havainnointiin ja robottien ohjaamiseen esteiden ohi.



Kuva 11. Kinectillä varustettu lentävä robotti [30]

Kuvan 11 robotti on projekti Berkeleyn yliopistosta Kaliforniassa. Se osaa lentää omin avuin ja vältellä kappaleita Kinectiltä saadun syvyyskuvan avulla. Samasta syvyystiedosta voidaan päätellä myös lentokorkeus.

Ympäristön havainnointi perustuu täysin avoimen lähdekoodin ajureihin. Prosessointi tapahtuu 1,6 gigahertsin Intel Atomiin pohjautuvassa koneessa, jolla pyörii Ubuntun versio 10.04. Laite on siis reilusti alle Xbox 360 -konsolin tehojen mutta kykenee silti käsittelemään sensorin syvyyskuvaa suurella tarkkuudella.

Kinect-sensoria voisi siis esimerkiksi käyttää automatisoidun imurirobotin navigoinnissa tai pienemmissä sovelluksissa ja laitteissa, joiden täytyy havainnoida etäisyyttä kappaleisiin. Hintaansa nähden Kinectin sensori on kuitenkin melko kallis ja ympäristön havainnointiin on olemassa muita halvempia ratkaisuja. [30.]

4 Sovelluskehitys Kinectille

Kinectille ei ole julkaistu vielä valmiita virallisia kehitystyökaluja, joten kehityksessä pitää käyttää joko avoimen lähdekoodin ajureita tai Microsoftin beta-asteella olevia kehitystyökaluja. Virallinen työkalupaketti on luvattu joskus tulevaisuudessa, mutta sille ei ole annettu tarkkaa päivämäärää eikä tietoa siitä, mitä kaikkia alustoja se tukee.

Kehityskoneelta vaaditaan suorituskykyä sekä USB 2.0 -liitäntä. Käyttöönotossa tulee huomioida, että samaan USB Host Controlleriin on liitetty ainoastaan Kinect eikä muita laitteita. Kinect vaatii itselleen kaiken käytettävissä olevan kaistan, koska tietoa liikkuu valtava määrä. On myös huomioitava että laite on suunniteltu konsolin peliohjaimeksi, eikä käytettäväksi PC:n yhteydessä. Tästä syystä tietokone voi käyttäytyä arvaamattomalla tavalla laitetta poistaessa. Testikäytössä koko kehityskone sammui irrottaessa Kinectin virtalähteen vielä kun USB-kaapeli oli yhdistettynä koneeseen. [17.]

4.1 Avoimen lähdekoodin ajurit

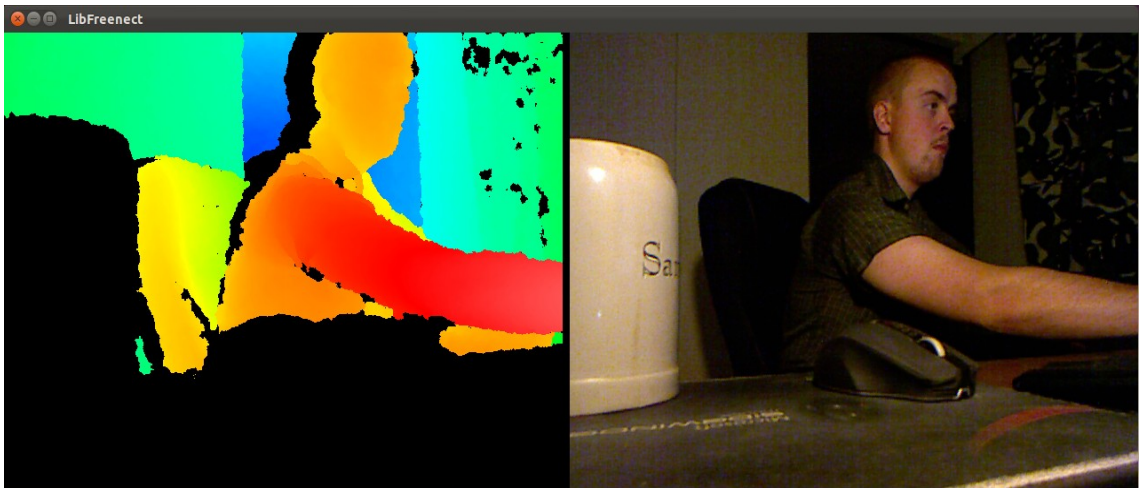
Avoimen lähdekoodin ajureihin liittyvä wiki löytyy osoitteesta OpenKinect.org. Virallisten kehitystyökalujen julkaisu ei ole tappanut kehitystyötä, vaan yhteisö on yhä hyvin aktiivinen. Testatessa asensin epäviralliset ajurit Ubuntun versioon 11.10 esimerkin 1 mukaan, eikä vastaan tullut yhtään ongelmia.

```
sudo apt-get install git-core cmake libglut3-dev pkg-config build-
      essential libxmu-dev libxi-dev libusb-1.0-0-dev
git clone https://github.com/OpenKinect/libfreenect.git
cd libfreenect
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig /usr/local/lib64/
```

Esimerkkikoodi 1. Ubuntun manuaalinen asennus [31]

Linuxin käyttöönotto tapahtuu erittäin nopeasti. Esimerkkiohjelmia on monenlaisia, ja ne tarjoavat heti pääsyn visualisoituun syvyyskuvaan, Kinectin ohjelmoitavaan lediin sekä moottoroituun jalustaan. Toisaalta on syytä muistaa, että Linuxin kanssa asiat eivät aina etene odotettuun tapaan.

Testikoneena toimi erittäin vanha 1.6 GHz AMD Turion ML-28 -prosessorilla varustettu kannettava, joka kykeni silti hyödyntämään sensoria melko hyvin. Esimerkkiohjelmat toimivat sulavasti, mutta on huomioitava, ettei yhdessäkään niistä tehdä muuta kuin käsitellään tietovirtoja ja visualisoidaan syvyyskuva. Jo pelkästään syvyyskuvan värjääminen syvyyden havainnollistamiseksi vaatii huomattavan määrän tehoa.

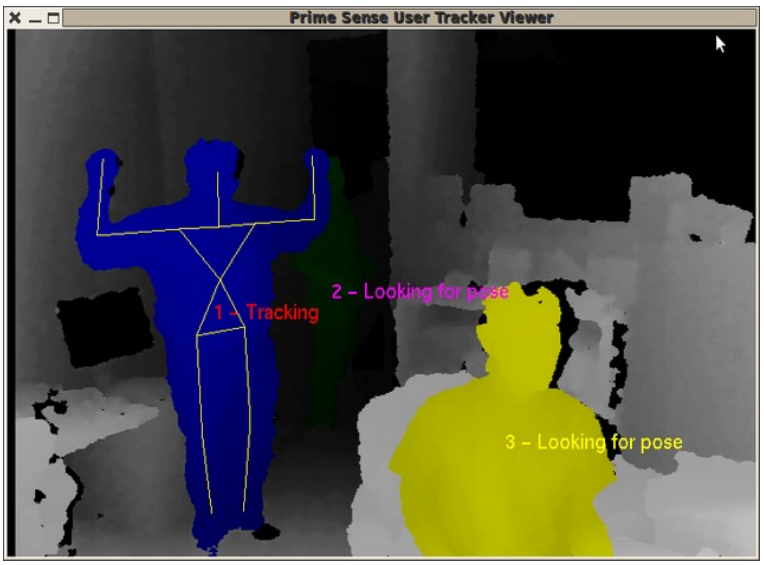


Kuva 12. Syvyyskuvan visualisointi esimerkissä

Kuvan 12 esimerkkiohjelma glview kertoo heti, toimiiko laite oikein. Se näyttää tavallisen kameran ulosannin sekä syvyyskuvan, joka on visualisoitu niin, että värit kulkevat punaisesta siniseen, missä punainen on lähimpänä, sininen kauimpana. Sillä voi myös vaihtaa Kinectin LED-valon tilaa ja kallistaa sensoria ylös ja alas. Mitään tarkempaa pelaajatunnistusta se ei kykene suorittamaan.

Kuva 13. Syvyyskuva ja värikuva yhdistettynä

Ajurien mukana tuli myös tämä mainitsemisen arvoinen esimerkki, jossa syvyyskuvaan on yhdistetty myös värikameran ulosanti. Kuvassa 13 näkyy ylävartalon muodostama sokea alue sekä avoin ympäröivä huone kolmiulotteisena mallina. Syvyystietoa ei käytetä pisteen visualisoimiseen eri välillä vaan pisteen sijoittamiseen kolmiulotteiseen malliin. Värinä toimii kamerasta tullut tieto. Kolmiulotteista mallia voi pyöritellä ja siitä näkee käytännössä heti kaiken, mitä sensori voi kahdella kamerallaan havaita.



Kuva 14. PrimeSensen luurankomallinnus (OpenNI)

Esimerkit ovat kattavia, ja kehitystyön voi aloittaa saman tien C/C++:llä. Ohjelmatiedostoon tarvitsee ainoastaan lisätä avoimen lähdekoodin kirjastot. Tässä törmää kuitenkin heti isoimpaan epävirallisten ajurien ongelmaan eli siihen, ettei mitään kirjastoja tule valmiina. Omalla osaamisella ja taidolla syvyyskuvasta voi saada irti vaikka mitä.

Mikäli luurankotunnistuksen haluaa saada toimimaan avoimen lähdekoodin ajureilla, hyvä lähtökohta on OpenNI API, jolla saa käyttöön luurankotunnistuksen, kuten kuvassa 14. Kyseessä on avoimen lähdekoodin kirjasto, jossa on yksinkertaisten eleiden tunnistamista ja koko vartalon seurantaa. OpenNI -organisaatio perustettiin marraskuussa 2010, ja yksi sen keskeisistä perustajista on Kinectin teknologian takana oleva PrimeSense. [32.]

4.2 Microsoft Beta SDK

Toistaiseksi virallinen Kinect SDK (Software Development Kit) on julkaistu ainoastaan Windows 7 -alustalle 32- ja 64-bittisenä versiona. Syy rajalliseen tukeen on se, että kehitysryhmä on pieni ja vanhempiin käyttöjärjestelmiin keskittyminen veisi resursseja muulta kehitystyöltä. Tukea ei ole toistaiseksi luvattu muille järjestelmille, joten vanhempaa Windowsia käyttävä henkilö joutuu turvautumaan epävirallisiin ajureihin. Koska kyseessä on reaaliaikainen syvyyskuvan käsittely, laitteistolta vaaditaan huomattavasti tehoa. Kehityskoneelta vaatimuksiksi esitetään:

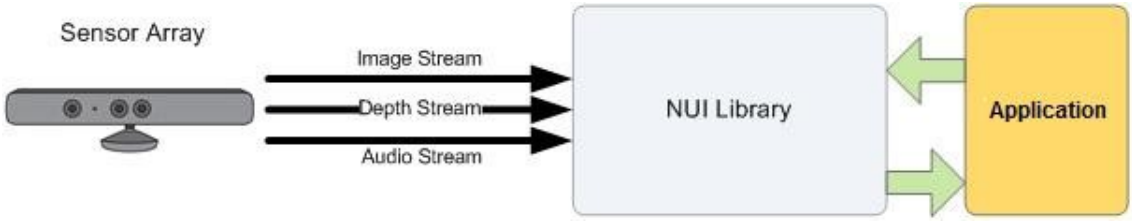
Laitteiston minimivaatimukset

- 2,66 -Ghz:n tuplaydinprosessori tai nopeampi
- Windows 7– yhteensopiva näytönohjain joka tukee DirectX® 9.0c -ominaisuuksia
- 2 GB muistia (4 GB RAM suositeltu)

Ohjelmistopuolen vaatimukset

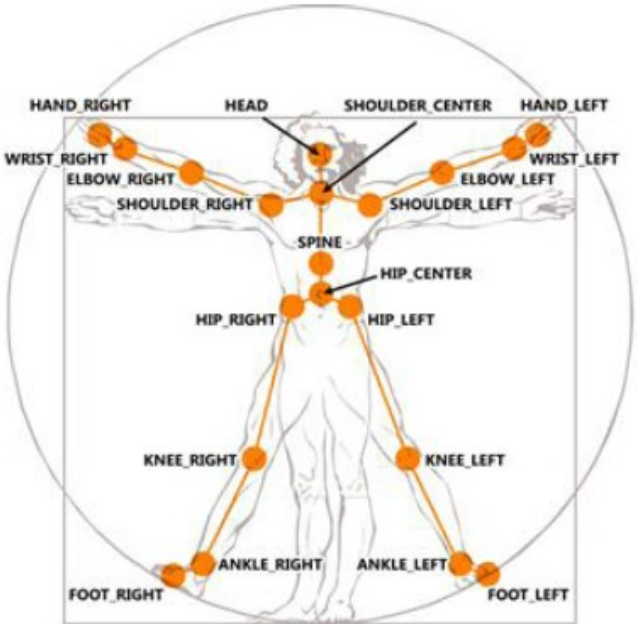
- Visual Studio 2010 Express (tai muu 2010 -versio)
- Microsoft .NET Framework 4.0

Osa esimerkeistä vaatii myös muita paketteja varsinkin puheen tunnistamiseen. Näitä paketteja ovat Microsoft Speech Runtime ja SDK sekä englanninkielen akustinen malli.



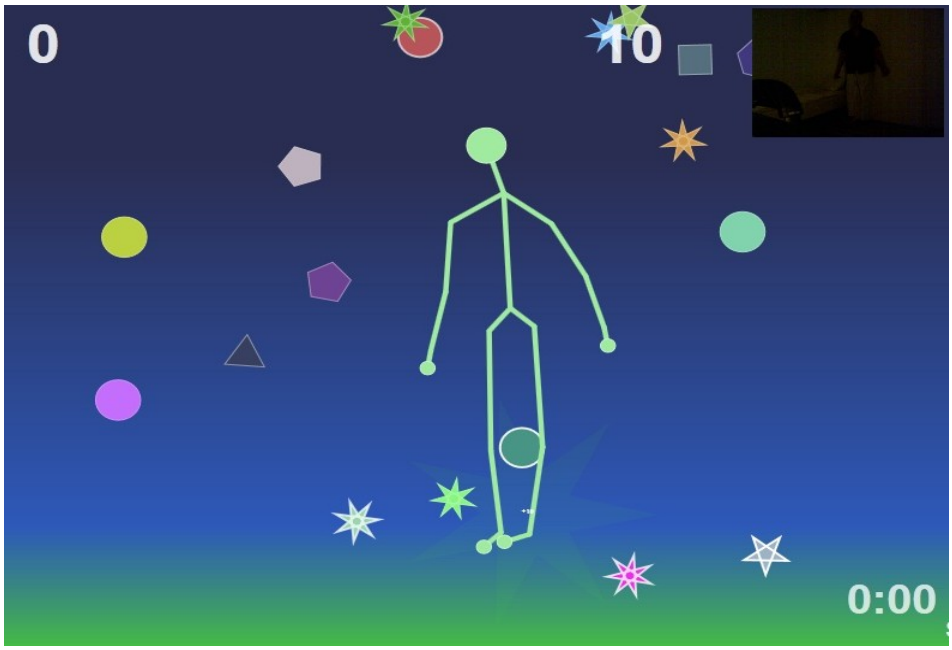
Kuva 15. Yksinkertaistettu malli kehityskirjaston rakenteesta. [17, s.14.]

Kuvan 15 NUI -kirjasto (Natural User Interface) sisältää luurankotunnistuksen ja ääni-ominaisuudet. Itse puheentunnistus on integroitu Microsoft Speechiin. NUI -kirjasto toimii niin, että se tulkitsee laitteelta tulleet tietovirrat ohjelmalle helposti käytettävään muotoon. Tehoja kirjasto syö paljon, ja esimerkkiprojekteilla oli helppoa saada kone hyytymään, vaikka testikone oli vaatimuksia tehokkaampi.



Kuva 16. Luurankomallin pisteet suhteessa ihmisvartaloon [17, s.20.]

Luurankotunnistus perustuu 20 eri vartalon pisteeseen kuvan 16 mukaan. Kirjastoissa tulee myös tuki kahden luurangon seurantaan, ja se kykenee ennakoimaan raajojen liikettä, mikäli jostakin syystä sensori ei sillä hetkellä näe kyseistä osaa pelaajasta. Se on kuitenkin melko raskas laskentaoperaatio, ja sen vaikutus on havaittavissa heti.



Kuva 17. Kinect SDK ShapeGame

Kehitystyökalujen mukana tullut esimerkkipeli ShapeGame (kuva 17) tarjoaa hyvän näytteen siitä, mitä kaikkea Kinectillä voi tehdä. Pelin idea on huitoa taivaalta valuvia kappaleita, ja siitä löytyy esimerkki kaksinpelistä kahdella luurangolla. Ääniohjauksella voi nopeuttaa tai hidastaa peliä, vaihtaa tietynväristen kuvioiden kokoa tai keskeyttää koko pelin. Ohjaus toimii niin kauan, kun puhuu haluttuja sanoja, mutta jos puhuu mitä sattuu, huomaa heti, että ohjelma tunnistaa täysin järjettömästäkin puheesta virheellisesti sanoja.

4.3 Vertailu

Epäviralliset ajurit ja Microsoftin viralliset kehitystyökalut ovat suurin piirtein yhtä helppo ottaa käyttöön. Molemmissa on loistavia esimerkkejä siitä, mitä Kinectillä voidaan saavuttaa. Paperilla viralliset kehitystyökalut ovat ominaisuuksiltaan paljon paremmat, mutta koska ne ovat saatavilla vain yhdelle alustalle, monelle ei ole muuta vaihtoehtoa kuin avoimen lähdekoodin ajurit.

Avoimen lähdekoodin ajureiden yhteydessä luurankotunnistukseen tarvitsee OpenNI:n, jossa luurankoseurannan tarkkuus on suurin piirtein samaa tasoa kuin Microsoftin luu-

rankoseuranta. Toisaalta Microsoftin kehitystyökalujen luurankotunnistus osaa myös ennakoida missä raaja on, mikäli palautetta sensorilta ei saada juuri sillä hetkellä. Tätä ominaisuutta ei löydy OpenNI:n luurankoseurannasta. Avoimen lähdekoodin ajureihin pohjautuvissa sovelluksissa on se etu, että sovelluksia voi viedä kaupalliseen käyttöön, toisin kuin virallisilla kehitystyökaluilla, joihin ei ole kaupallista lisenssiä vielä. Yhtään kaupallista sovellusta ei toisaalta ole nähty, ja Microsoftin reaktio näihin sovelluksiin on osaltaan spekulointia.

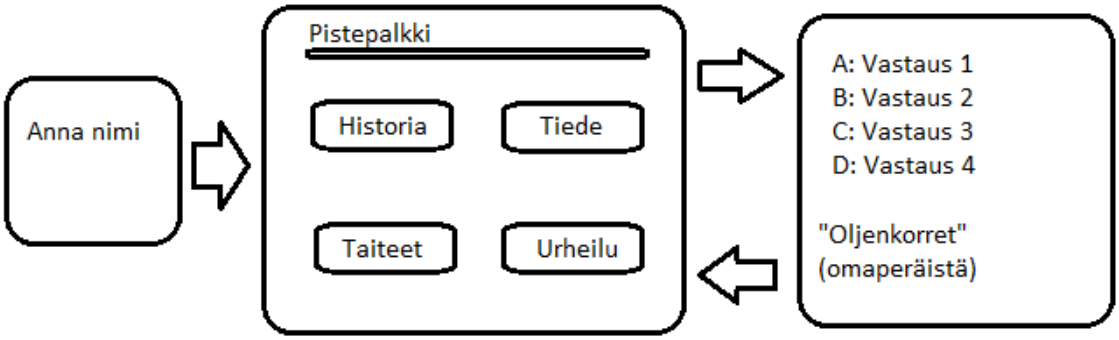
Mikäli laitteella haluaa saada jotakin tehtyä nopeasti, Microsoftin kehitystyökalut ovat ehdoton valinta, jos vaan on käytössä Windows 7 -käyttöjärjestelmä. Kaikkia ominaisuuksia ei ole vielä avoimen lähdekoodin ajureissa, mutta virallisessa SDK:ssa on kaikki tarvittava. [33.]

5 Esimerkkisovellus

Insinöörityön päämääränä oli kehittää lopulta sovellus, joka hyödyntää mahdollisimman paljon Kinectin ominaisuuksia. Koska kyseessä on melko monimutkaista teknologiaa, on myös tärkeä löytää aihe, joka kokonsa puolesta pysyy hallittavana eikä koidu tekniseltä osaamisvaatimuksilta liian haastavaksi. Sisällön tuottamisen tulee olla nopeaa ja helppoa, joten on vältettävä kolmiulotteista tai käsin piirrettävää grafiikkaa. Ohjelmalla tulee olla myös oikea käyttötarkoitus, jotta se ei ole pelkkä teknologiademo.

5.1 Idean esittely

Koska Kinect alkujaan suunniteltiin pääasiallisesti peliohjaimeksi, esimerkkisovelluksena toimii yksinkertainen tietovisa, joka käyttää liikkeen- ja puheentunnistusta. Ohjelmallisesti se on helppo toteuttaa eikä vaadi monimutkaista grafiikkaa, mikä ei itsessään ole oleellinen osa tätä työtä. Tärkeää on myös se, että ohjelmalla on oikea käyttötarkoitus ja tämä selkeä kokonaisuus pakottaa suunnittelemaan ohjelman rakennetta, mikä on olennainen ohjelmistosuunnittelijan taito.



Kuva 18. Yksinkertainen visio pelistä

Pelin pelkistetty rakenne näkyy kuvassa 18. Alkuruudulta on mahdollista valita kysymystyyppi joko liike- tai äänentunnistusta hyödyntäen. Seuraavalta ruudulta valitaan vastaus samalla ohjaustavalla. Mikäli kysymys on liian vaikea, pelaajalla on käytössä erilaisia "oljenkorsia", joilla voi selvitä tilanteesta. Kysymykset noudetaan jonkinlaisesta tietokannasta tai tietorakenteesta, jota voi muokata ulkopuolisella ohjelmalla vapaasti koskematta lähdekoodiin. Peli pysähtyy viiden kysymyksen jälkeen ja pistelaskennassa otetaan huomioon pelaajan nopeus vastattaessa.

Tietovisa kattaa sopivasti Kinectin ominaisuudet eli käyttää liike- ja ääniohjausta. Kokonsa puolesta se sopii mainiosti esimerkkisovellukseksi, eikä sisällön tuottamiseen mene liikaa aikaa.

5.2 Toteutus

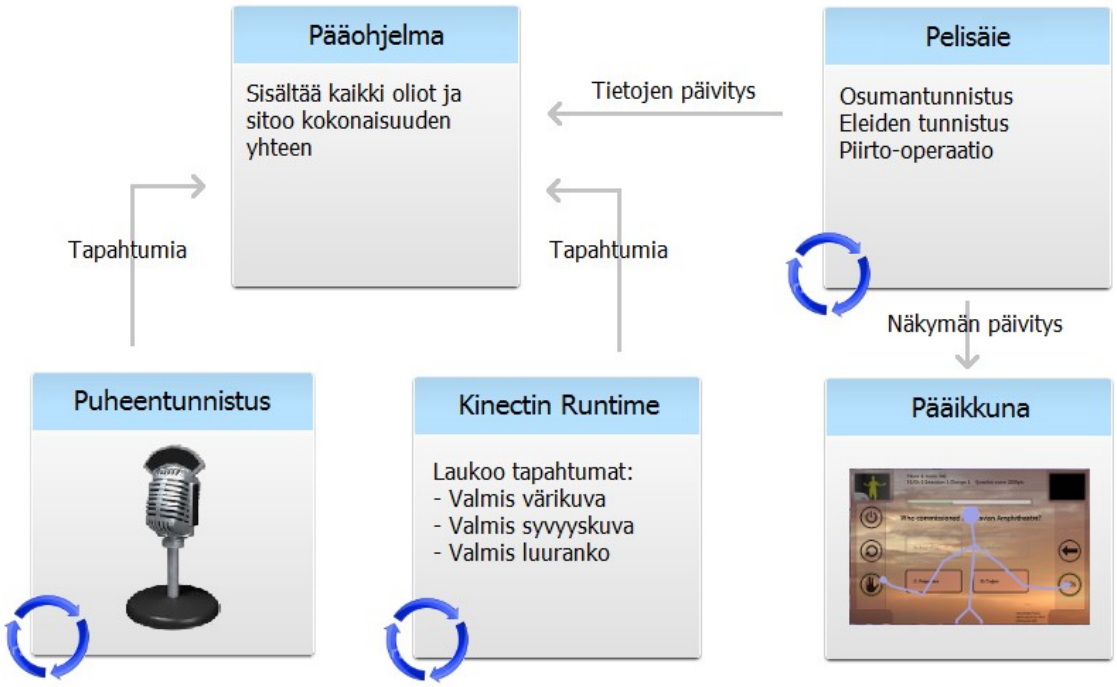
Sovellus toteutettiin Microsoftin beta-asteella olevilla kehitystyökaluilla, koska niillä pääsee heti hyödyntämään luurankomallia eli saa käyttöön jonkinlaisen ohjaustavan ilman suurempaa vaivaa. Avoimen lähdekoodin ajureilla syvyyskameran palautteen tulkitseminen siihen muotoon, että sitä voisi käyttää edes jotenkin järkevästi sovelluksen hallitsemiseen vaatii huomattavasti enemmän työtä. Virallisten kehitystyökalujen mukana tulevat esimerkit ovat hyvin selitetty ja niistä on helppo aloittaa oma projekti. Esimerkkejä on laadittu kahdella eri kielellä, C++:lla ja C#:lla. Kaikkein kattavin niistä eli kokonainen peli ShapeGame, joka hyödyntää kaikkia Kinectin ominaisuuksia, on ainoastaan C#:lla kirjoitettuna. Sovelluksen kieli on siis C# esimerkkipelin ja myös sen takia, että

se on entuudestaan tuntematon kieli. Ohjelmointityökaluna toimii Microsoftin Visual Studio 2010 Ultimate. Dokumentaatio on myös kiitettävällä tasolla ja käyttöönotosta on tehty useita tutoriaalivideoita koskien eri aihepiirejä kehitystyökaluista. [34.]

Virallisten työkalujen mukana tulee myös täysi tuki Kinectin eri ääniominaisuuksille ja esimerkki siitä, miten niitä tulee käyttää. Kielipaketti on ainoastaan englanninkielisenä, joten sovellus ei ole suomenkielinen. Peli lataa kysymykset yksinkertaisesta paikallisesta SQL-tietokannasta. Kysymyksille olennaisia tietoja on vaikeus, kategoria ja yksilöivä tunnus, joka takaa, että samaa kysymystä ei tule kahdesti.

5.3 Ohjelman rakenne

Ohjelmassa on kolme itsenäistä säiettä, joissa ajetaan puheentunnistusta, Kinectin ominaisuuksia sekä pelin toiminnallista logiikkaa. Pelin pääohjelma vastaanottaa tapahtumia puheentunnistajalta ja Kinectiltä. Pelisäikeessä päivitetään näkymää, tarkistetaan pelaajan liikkeet, vertaillaan niitä eleisiin sekä nappuloihin, ja poistetaan pelialueelta poistuneet pelaajien luurangot ohjelmasta. Kaavio 1 esittää toiminnan pelkistettynä.



Kaavio 1. Ohjelman rakenne

5.3.1 Luokkamäärittelyt

Ohjelma on jaettu 17 eri luokkaan. MainWindow-luokka toimii ohjelman aloituspisteenä ja sen sisältä löytyy varsinainen pelisäie, joka ohjaa vastuun pääluokalle MillionaireGame, jossa varsinainen pelilogiikka on. Peliruutu, jolle piirretään, on myös MainWindow:n sisällä, ja pääluokalla on vapaa pääsy siihen. Syy erikoiseen ratkaisuun oli yksinkertaisesti se, että en saanut ohjelmaa toimimaan muulla tavalla. Kaaviossa 2 näkyy pelin tärkeimmät luokat.

Kaavio 2. Toiminnalle tärkeimmät luokat

5.3.2 Graafinen toteutus

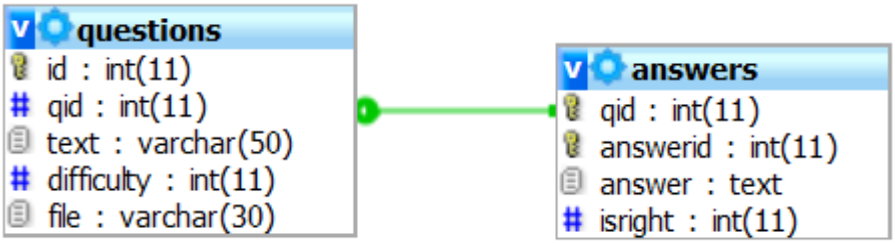
Tietovisa rakentuu esimerkkipelin pohjalle, joten siinä käytetään Windows Presentation Foundationia (WPF). Se on .NET Frameworkin kirjasto, joka muodostaa Windowsin graafisen rajapinnan. WPF toimii käyttöliittymän pohjana ja grafiikan, tekstin, animaatioiden ja videon piirron rajapintana. Ohjelmaruutu voidaan alustaa erillisessä XML-tiedostossa, joka on tyyppiä xaml. Visual Studiosta löytyy suunnittelutyökalu, jolla voi sijoitella elementtejä vapaasti ja muokata ruudun kokoa. Ruutu on lukittu tiettyyn resoluutioon eikä sitä voi muokata ajonaikaisesti mitenkään. Ohjelma ei kuitenkaan käytä mitään valmiita WPF:n käyttöliittymäelementtejä nappuloille eikä sitä ole rakennettu Visual Studion suunnittelutyökalulla. Syy tähän oli se, että omat elementit oli helpompi säistää ja pitää yksinkertaisina. [35.]

Sen sijaan jokainen näkyvä ruutu toteuttaa abstraktin luokan InteractiveScreen, josta löytyy abstrakti metodi piirtämiseen, komennon käsittelemiseen sekä ruudun sanakirjaston noutoon. Jokainen näkyvä ruutu on siis oma olionsa. Pääohjelmassa on Interac-

tiveScreen-tyyppinen olio, johon sijoitetaan aina käytössä oleva ruutu. Piirto-operaatiossa tälle ruudulle annetaan pääsy näkyvään peliruutuun, joka on WPF Canvas -olio, johon ikkunaolio sijoittaa omat käyttöliittymäelementtinsä ja kuvionsa.

5.3.3 Kysymykset

Kysymykset tuodaan pienestä tietokannasta joka on toteutettu SQLite- relaatiotietokantajärjestelmällä. Se on erittäin pieni ja kevyt, eikä siihen tarvita erillistä tietokannanhallintaohjelmaa tai palvelinta. Sen käyttöönottoon C# -kielisessä ohjelmassa on olemassa avoimen lähdekoodin kirjasto, jonka linkitin projektiin. [36; 37.]



Kaavio 3. Tietokannan yksinkertainen rakenne

Tietokannan rakenne on esitetty kaaviossa 3, ja se on rakenteeltaan erittäin yksinkertainen. Kysymystaulusta löytyvät perustiedot, kuten vaikeus, kysymyskategoria sekä yksilöivä tunnus. Alkuperäisestä suunnitelmasta poiketen taulussa on myös ylimääräinen kenttä, joka määrittelee tiedostonimen, jos vaikkapa musiikkikysymyksissä halutaan soittaa kappale, joka pitää tunnistaa tai kuvataidekysymyksissä näytetään maalaus. Vastaustaulussa kysymyskategoria ja vastauksen oma tunnus muodostavat pääavaimen, jotta vastaukset eivät voi mitenkään mennä väärille kysymyksille.

Esimerkkisovelluksen tavoite on testata Kinect-sensoria, joten virhetilanteisiin reagointi ei ole etusijalla ohjelman toiminnassa. Kysymyksiä noudettaessa oletetaan, että niitä tulee aina neljä kappaletta ja niistä löytyy oikea vastaus. Ohjelmasta ei löydy minkäänlaisia turvatarkistuksia niille tapauksille, joissa tämä ei pidä paikkaansa. Liika kysymysmäärä ei kuitenkaan kaada sovellusta vaan ruudulle piirtyvät kaikki vastaukset, joista osa menee pakosti näkyvän alueen ulkopuolelle.

```
public static Question GetQuestion(Question.Type qType, int difficulty){
	string query = "SELECT \"text\",\"answer\",\"isright\",\"file\"  FROM
			questions INNER JOIN answers " +
			"ON questions.id = answers.qid " +
			"WHERE questions.qid = "+(int)qType+" AND difficulty =
			"+difficulty+";";
	DataTable dt = new DataTable();
	try {
		String dbConnection = "Data Source=questions.s3db";
		SQLiteConnection cnn = new SQLiteConnection(dbConnection);
		cnn.Open();
		SQLiteCommand mycommand = new SQLiteCommand(cnn);
		mycommand.CommandText = query;
		SQLiteDataReader reader = mycommand.ExecuteReader();
		dt.Load(reader);
		reader.Close();
		cnn.Close();
	}
	catch (Exception e) {
		throw new Exception(e.Message);
	}
}
```

Esimerkkikoodi 2. Kymysten nouto tietokannasta

Kysymykset noudetaan yksinkertaisella esimerkkikoodista 2 löytyvällä hakulauseella vaikeusasteen ja kategorian mukaan. Tietokanta on yksittäinen paikallinen tiedosto. Typistetystä koodiesimerkistä ei käy ilmi, miten niistä muodostetaan Question-olio, joka annetaan ulos metodista muodostettavalle kysymysruudulle.

5.4 Kinectin hyödyntäminen ohjelmassa

Aloitin sovelluksen rakentamisen työkalujen mukana tulevien esimerkkien pohjalta, joten pääsin pienten alkuvaikeuksien jälkeen käsiksi johonkin konkreettiseen. Olennaiset paketit ohjelman toimimiselle ovat Microsoft.Research.Kinect ja Microsoft.Speech. Jälkimmäistä ei tarvitse, mikäli ei halua käyttää äänitunnistusta ollenkaan. Molemmat jaetut kirjastot on lisättävä luotuun projektiin.

5.4.1 Kinectin alustaminen

Ennen kuin sensorista saa mitään ulos, se tulee käynnistää halutuilla parametreilla. Ohjelmassa ei ole minkäänlaista reagointia siihen, löytyykö laitteistosta Kinectiä vai ei,

koska ohjelma ei muutenkaan voisi toimia mitenkään ilman laitetta. Tässä vaiheessa ongelmaan voi kuitenkin reagoida ja siirtää ohjauksen vaikkapa hiiripohjaiseksi.

```
public void Initialize(){
	Runtime nui = new Runtime();
	try{
		nui.Initialize(RuntimeOptions.UseDepthAndPlayerIndex |
		RuntimeOptions.UseSkeletalTracking | RuntimeOptions.UseColor);
	}
	catch (Exception _Exception){
		Console.WriteLine(_Exception.ToString());
		return false;
	}
	nui.DepthStream.Open(ImageStreamType.Depth, 2,
		ImageResolution.Resolution320x240, ImageType.DepthAndPlayerIndex);
	nui.VideoStream.Open(ImageStreamType.Video, 2,
		ImageResolution.Resolution640x480, ImageType.Color);
	nui.SkeletonEngine.TransformSmooth = true;
	nuiInitialized = true;
}
```

Esimerkkikoodi 3. Kinectin alustaminen

Kinectiä alustaessa sille kerrotaan, mitä ominaisuuksia halutaan käyttää. Esimerkkikoodissa 3 syvyyskuva on rajattu pienempään resoluution kuin laite pystyy, koska resoluution kasvaessa kasvaa myös tarvittava laskentateho. Isompaan resoluutioon pääsee käsiksi, jos vaihtaa avaamisessa annetuksi parametriksi pelkän syvyyden ilman pelaajaindeksiä. Jos resoluutiota yrittää nostaa suoraan pelaajaindeksin ollessa päällä, kaatuu koko sovellus.

5.4.2 Luurankotunnistus

Peli ei varsinaisesti tarvitse koko luurankoa toimiakseen, ainoastaan käsipisteet. Koko vartaloa seurataan kuitenkin, koska eleet tarvitsevat tietoa muista pisteistä. Samalla voidaan myös piirtää koko pelaaja ruudulle, mikä helpottaa pelaajan käyttämän alueen havainnointia. Koko luurangon käyttämisestä löytyi hyvät esimerkit kehitystyökalujen mukana tulleesta pelistä, jossa kaksi pelaajaa voi huitoa taivaalta tippuvia kappaleita.

```
nui.VideoFrameReady += new EventHandler<ImageFrameReadyEventArgs>
	(nui_ColorFrameReady);
nui.SkeletonFrameReady += new EventHandler<SkeletonFrameReadyEventArgs>
	(nui_SkeletonFrameReady);
```

```
nui.DepthFrameReady += new EventHandler<ImageFrameReadyEventArgs>
       (nui_DepthFrameReady);
```

Esimerkkikoodi 4. Tapahtumankäsittelijät

Kinectiä alustettaessa esimerkkikoodissa 4 linkitetään pääohjelmassa sijaitsevat tapahtumankäsittelijät Runtime-olioon. Ohjelmassa on käytössä kaikki ominaisuudet, joten syvyys-, väri- ja luurankokehyksestä tulee oma tapahtuma. Kun pelaaja tunnistetaan tai pelaajan liikettä päivitetään, kutsutaan tapahtumankäsittelijää, jolle viedään parametrina saatu luurankomalli.

```
void nui_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e){
            SkeletonFrame skeletonFrame = e.SkeletonFrame;
            int iSkeletonSlot = 0;
            foreach (SkeletonData data in skeletonFrame.Skeletons){
                if (SkeletonTrackingState.Tracked == data.TrackingState){
                    PlayerSkeleton player;
                    if (players.ContainsKey(iSkeletonSlot)){
                           player = players[iSkeletonSlot];
                    }
                    else if (playersAlive > 0) {
                           pdata[1].skeletonID = iSkeletonSlot;
                           player = new PlayerSkeleton(iSkeletonSlot,
                           PlayerSkeleton.PlayerColor.Blue);
                    }
                    else{
                           pdata[0].skeletonID = iSkeletonSlot;
                           player = new PlayerSkeleton(iSkeletonSlot,
                           PlayerSkeleton.PlayerColor.Red);
                    }
                    players.Add(iSkeletonSlot, player);
                    }
                    player.lastUpdated = DateTime.Now;

                    if (data.Joints.Count > 0){
                       player.isAlive = true;
                       player.UpdateJointPosition(data.Joints, JointID.Head);
                       player.UpdateJointPosition(data.Joints,JointID.HandLeft);
                       player.UpdateJointPosition(data.Joints,JointID.HandRight);
                    }
                }
                iSkeletonSlot++;
            }
        }
```

Esimerkkikoodi 5. Tapahtuma valmiille luurankokehykselle

Kun luurankokehys on valmis ja tästä viedään tieto tapahtumankäsittelijälle esimerkkikoodissa 5, tarkistetaan, onko pelaaja jo olemassa ruudulla. Mikäli pelaajaa ei ole, luodaan uusi pelaaja. Sillä tarkoitetaan seurattavaa luurankoa, joka voi tuhoutua millä hetkellä hyvänsä. Tästä syystä itse pelaajan pelitiedot, kuten pisteet ja jäljellä olevat oljen-

korret, on pidettävä turvassa erillisessä säiliöluokassa, johon luuranko linkitetään yksilöivän tunnuksen mukaan. Tämän tarkistusvaiheen jälkeen päivitetään jokainen luu ja jokainen piste luurangosta. Koodiesimerkissä se on typistetty kolmeen riviin.

```
public void UpdateJointPosition(Microsoft.Research.Kinect.Nui.JointsCollection
       joints, JointID j) {
       var seg = new Segment(joints[j].Position.X * playerScale + playerCenter.X,
                                 playerCenter.Y - joints[j].Position.Y *
                                 playerScale, joints[j].Position.Z, j);
       seg.radius = playerBounds.Height * ((j == JointID.Head) ? HEAD_SIZE :
                    HAND_SIZE) / 2;
       UpdateSegmentPosition(j, j, seg);
}
```

Esimerkkikoodi 6. Luuranko-olion sisällä tapahtuva päivitys

Luuranko-olion sisältä löytyy metodi yksittäisen pisteen sekä luun päivittämiseen. Esimerkissä 6 päivitetään pisteet, joita ovat molemmat kädet, jalat sekä pää. Kertomalla vain yksi piste, tehdään piirtovaiheessa ero siihen, millä tavalla kyseinen kohta pitäisi piirtää.

5.4.3 Kosketeltavat nappulat

Ohjelman kaikki kosketukseen reagoivat nappulat ovat saman luokan olioita ja niihin on linkitetty komento, joka ajetaan pelaajan painaessa nappulaa. Painallus toimii niin, että pelaajan viedessä vasemman tai oikean käden nappulan päälle, siitä otetaan talteen osumissyvyys, jota vertaillaan käden liikkeeseen alueen sisällä. Jos käsi siirtyy tarvittavan etäisyyden lähemmäksi Kinectiä pysyessään nappulan sisällä, painallus rekisteröityy. Mikäli käsi lähtee pois nappulan alueelta, vanha osumissyvyys poistuu eikä vertailua suoriteta enää.

```
public bool CheckHit(Point3D refPoint, JointID jId){
            if (!isDisabled && withinScreen(refPoint)){
                if (this.bounds.Contains(new Point(refPoint.X, refPoint.Y))){
                    if (this.isSelected){
                        double hitDepth;
                        if (hits.TryGetValue(jId, out hitDepth)){
                            return ((hitDepth - refPoint.Z) > 0.15);
                        }
                    }
                    else{
                        this.isSelected = true;
                        hits.Add(jId, refPoint.Z);
                        clickSound.Play();
                    }
```

```
                }else
                {
                    hits.Remove(jId);
                    if (hits.Count == 0){
                        this.isSelected = false;
                    }
                }
            }
            return false;
        }
}
```

Esimerkkikoodi 7. Osumantunnistus nappuloissa

Osumantunnistus ei ole rajattu pelkästään käsille, vaan sitä voi käyttää mihin tahansa luurankomallin pisteeseen, jonka määrittää esimerkkikoodissa 7 luurankopisteen yksilöivä tunnus JointID. Osuman tapahtuessa laitetaan nappulaoliossa sijaitsevaan listaan talteen raajan yksilöivä tunnus sekä osumissyvyys. Tieto osumasta säilyy listassa niin kauan kuin osuman antanut luurankopiste on rajatulla alueella. Kun valittu luurangon piste liikkuu tarvittavan etäisyyden lähemmäs, CheckHit-metodi palauttaa arvon tosi, johon pääohjelma reagoi halutulla tavalla.

5.4.4 Puheentunnistus

Ohjelmassa jokaisella interaktiivisella ruudulla on oma sanakirjastonsa, joista jokaiseen käskyyn on linkitetty samanlainen komento kuin nappuloihin. Äänikomennon toteutuessa käsky ajetaan samalla tavalla kuin se ajetaan nappulaa painettaessa. Tunnistusta ajetaan omassa säikeessään AudioRecognition-luokan sisällä. Kielistä käytössä on vain englanti, koska toistaiseksi kehitystyökaluille on olemassa ainoastaan englanninkielinen akustinen malli.

```
Dictionary<string, WhatSaid> GameplayPhrases = new Dictionary<string, WhatSaid>()
        {
            {"Game Restart", new WhatSaid()         {commands=Commands.Restart}},
            {"Game Reset", new WhatSaid()        {commands=Commands.Restart}},
            {"Game Exit", new WhatSaid()            {commands=Commands.Exit}},
            {"Game Quit", new WhatSaid()            {commands=Commands.Exit}},
         };

public void loadGrammar(Dictionary<string,WhatSaid> dict)
        {
       var choices = new Choices();
       foreach (var phrase in GameplayPhrases) { choices.Add(phrase.Key); }
       var actionGrammar = new GrammarBuilder();
```

```
        actionGrammar.AppendWildcard();
        actionGrammar.Append(choices);
        Grammar grammar = new Grammar(actionGrammar);
        sre.LoadGrammar(grammar);
}
```

Esimerkkikoodi 8. Käskykirjaston lataaminen

Lyhennetystä esimerkkikoodista 8 näkee, miten sanat ladataan tunnistajaan sanakirjastosta. Jokaiseen sanaan on liitetty sama komento kuin nappuloihin. Kyseinen esimerkki on itse äänentunnistajan sisältä ja komennot ovat pääkomentoja, joihin on laitettu etuliite sen takia, että saadaan minimoitua virheelliset tunnistukset, jotka voivat johtaa pahimmassa tapauksessa pelin sammumiseen.

```
RecognizerInfo ri = SpeechRecognitionEngine.InstalledRecognizers().
                Where(r => r.Id == RecognizerId).FirstOrDefault();
sre = new SpeechRecognitionEngine(ri.Id);
var t = new Thread(StartDMO);

private void StartDMO(){
        kinectSource = new KinectAudioSource();
        kinectSource.SystemMode = SystemMode.OptibeamArrayOnly;
        kinectSource.FeatureMode = true;
        kinectSource.AutomaticGainControl = false;
        kinectSource.MicArrayMode = MicArrayMode.MicArrayAdaptiveBeam;
        var kinectStream = kinectSource.Start();
        sre.SetInputToAudioStream(kinectStream, new SpeechAudioFormatInfo(
        EncodingFormat.Pcm, 16000, 16, 1, 32000, 2, null));
        sre.RecognizeAsync(RecognizeMode.Multiple);
}
```

Esimerkkikoodi 9. Puheentunnistuksen käynnistäminen

Puheentunnistus ei itsessään vaadi Kinectiä vaan sitä voi käyttää minkä tahansa mikrofonin kanssa. Kinectille voi kuitenkin määrittää, miten äänitystä tulee käyttää ja käytetäänkö lisäominaisuuksia, kuten akustisen kaiun poistamista. Esimerkkikoodissa 9 käytetään Kinectin ominaisuutta paikallistaa äänilähde. Tunnistusta käytettäessä tulee olla asennettua Microsoft Speech Platform Runtime ja SDK-paketit sekä englanninkielen akustinen malli.

```
sre.SpeechRecognized += sre_SpeechRecognized;
sre.SpeechHypothesized += sre_SpeechHypothesized;
sre.SpeechRecognitionRejected += sre_SpeechRecognitionRejected;

void sre_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
        {
            if (e.Result.Confidence > 0.96)
            {
```

```
            if (SaidSomething == null) { return; }

            var said = new SaidSomethingArgs();
            said.command = Commands.None;
            said.Phrase = e.Result.Text;

            List<Dictionary<string, WhatSaid>> allDicts = new
              List<Dictionary<string, WhatSaid>>() {GameplayPhrases,
              loadedDictionary};
            bool found = false;
            for (int i = 0; i < allDicts.Count && !found; ++i){
                foreach (var phrase in allDicts[i]){
                    if (e.Result.Text.Contains(phrase.Key)){
                        said.command = phrase.Value.commands;
                        found = true;
                        break;
                    }
                }
            }
            if (!found) { return; }
            SaidSomething(new object(), said);
            }
        }
}
```

Esimerkkikoodi 10. Sanojen tunnistaminen

Puheentunnistusta käynnistettäessä määritetään tapahtumankäsittelijät kolmelle eri tapahtumalle, eli hylkäämiselle, arvuuttelulle sekä hyväksymiselle. Tunnistuksen ollessa käynnissä puheesta arvuutellaan jatkuvasti mahdollisia sanoja, jotka löytyvät ohjelmaan ladatusta kirjastosta. Se ei tee eroa puheen tai muiden äänien välillä, joten käsien taputuksestakin yritetään päätellä, mitä sanaa se lähinnä muistuttaa. Kun tunnistin on lopulta sitä mieltä, että sana on tunnistettu, se kutsuu tapahtumankäsittelijää hyväksymistapahtumalle ja antaa sille olion, joka sisältää tunnistetun tekstinpätkän ja tiedon siitä, kuinka suurella varmuudella sana tunnistettiin. Esimerkkikoodissa 10 näkyy tapahtumankäsittelijä sanan hyväksymiselle. Tässä vaiheessa on oleellista tarkkailla varmuustekijää, koska ohjelma tunnistaa sanoja liiankin helposti. Kun varmuustekijä on tarpeeksi suuri, käydään läpi ohjelman sanakirjaa ja vertaillaan sitä tunnistettuun sanaan. Oikean sanan löytyessä otetaan siihen sanakirjastossa linkitetty komento, joka viedään pääohjelmalle käsiteltäväksi.

5.4.5 Eleiden tunnistaminen

Ohjelmaan oli tarkoitus kehittää ominaisuus, joka tunnistaa yksinkertaisia eleitä. Suuri haaste oli saada se toteutettua mahdollisimman yksinkertaisesti niin, ettei tarvitse mo-

nimutkaista matematiikkaa. Hankalaa tästä teki se, miten aloittaa tunnistus ja miten lopettaa se tai mihin verrata jatkuvasti muuttuvia kolmiulotteisia pisteitä. Päädyin ratkaisuun, jossa tunnistusta pyöritetään jatkuvasti ja pisteitä vertaillaan aina suhteessa pelaajan luurankomalliin. Eleet muodostuvat vapaasti määritellyistä alueista joita rajaa luurangosta valitut pisteet.

```
public static Gesture createCrossGesture(){
	Gesture gest = new Gesture(Commands.Halleluja);
	gest.addConstraint(new LocationConstraint(JointID.Head, 100,
		Constraint.Type.InsideCircle, 1500));
	gest.addConstraint(new LocationConstraint(JointID.ShoulderCenter, 100,
		Constraint.Type.InsideCircle, 1500));
	gest.addConstraint(new LocationConstraint(JointID.ShoulderLeft, 100,
		Constraint.Type.InsideCircle, 1500));
	gest.addConstraint(new LocationConstraint(JointID.ShoulderRight, 100,
		Constraint.Type.InsideCircle, 1500));
	return gest;
}
```

Esimerkkikoodi 11. Ristieleen luominen

Asioiden helpottamiseksi esimerkkieleet luodaan itse Gesture -olion sisällä staattisessa metodissa. Esimerkkikoodin 11 ele on ristinmerkki, jonka tehdessä ajetaan komento, joka soittaa Händelin Messiasta. Eleessä on lista rajoitteita, jotka voivat olla minkä tyyppisiä tahansa, kunhan toteuttavat vaaditun rajapinnan. Tässä tapauksessa käytetään rajoitetta, joka tarkkailee onko vertailukohteena oleva piste tietyn etäisyyden päässä toisesta pisteestä. Viimeisenä annettu arvo on aika millisekunteina, jonka aikana pelaajan on siirryttävä seuraavaan pisteeseen tai eleen tunnistaminen aloitetaan alusta. Vertailukohde on aina oikeanpuoleinen käsi, koska useamman pisteen seuranta on äärimmäisen hankalaa.

```
public override bool checkMatch(PlayerSkeleton p)
	{
	Point3D comparePoint = p.getJoint(compareJoint);
	 switch (type)
	{
		case Constraint.Type.InsideRect:
			Point3D p1 = p.getBone(j1);
			Point3D p2 = p.getBone(j2);
			Rect r = new Rect(Math.Min(p1.X, p2.X), Math.Min(p1.Y,
			p2.Y), Math.Abs(p1.X - p2.X), Math.Abs(p1.Y - p2.Y));
			return r.Contains(comparePoint.X, comparePoint.Y);
		case Constraint.Type.InsideCircle:
			Point3D p3 = p.getBone(j1);
			double X = Math.Abs(p3.X - comparePoint.X);
			double Y = Math.Abs(p3.Y - comparePoint.Y);
```

```
                    double dist = Math.Sqrt(X * X + Y * Y);
                    return (dist < circleRadius);
        }
        return false;
}
```

Esimerkkikoodi 12. Pisteen tunnistaminen

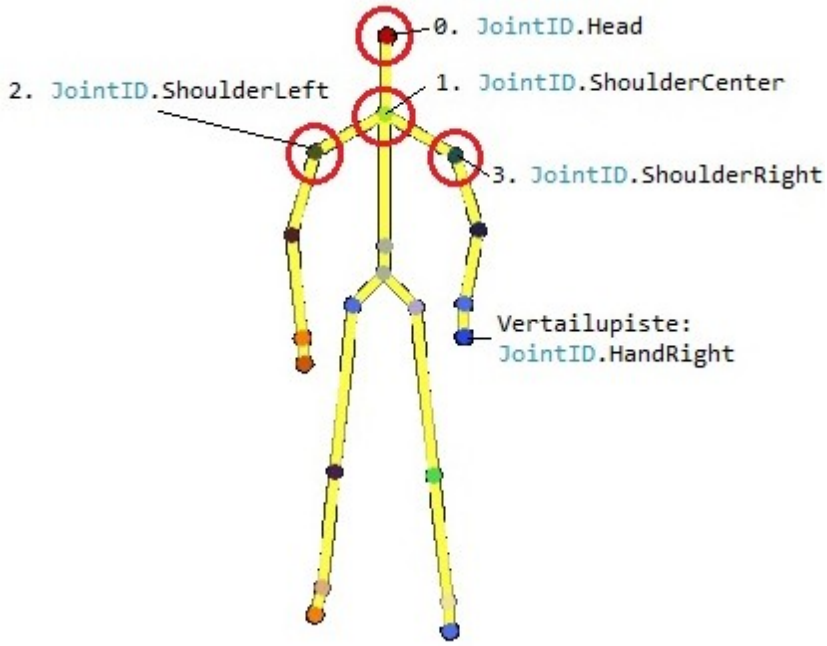
Pistettä tunnistettaessa on kaksi tapaa tunnistaa osuma. Koodiesimerkissä 12 näkyy metodi, joka tarkistaa eleen yksittäisen paikkarajoitteen. Ensimmäinen on pisteen tunnistaminen kahden pisteen muodostamasta neliöstä, joka lasketaan annettujen yksilöivien raajatunnusten avulla. Toinen tapa on yksinkertaisempi ja siihen tarvitaan ainoastaan yksi piste, joka muodostaa ympyrän, jonka alueella vertailupisteen on oltava. Vertailuoperaation tulos palautetaan suoraan, mihin eleen sisällä oleva metodi reagoi.

```
public Boolean checkMatch(PlayerSkeleton p){
if (constraints.Count == 0) { return false; }
            if (restart) {
                counter = 0;
                restart = false;
            }
            if (constraints[counter].checkMatch(p)){
                counter += 1;
                timer =
      DateTime.Now.AddMilliseconds(constraints[counter].TimeMs);
            }
            else if(counter != 0 && DateTime.Now > timer){
                restart = true;
            }

            if (constraints.Count == counter){
                restart = true;
                return true;
            }
}
```

Esimerkkikoodi 13. Siirto seuraavaan pisteeseen ja ajastus

Eleiden tunnistusta ohjaava logiikka on yksinkertainen ja perustuu laskuriin, jonka arvoa kasvatetaan aina käden käydessä halutulla alueella. Esimerkkikoodissa 13 on metodi, jonka kanssa pääohjelma kommunikoi. Laskuri toimii rajoitelistan indeksinä, eli sen arvon kasvaessa siirrytään vertailemaan käden liikettä seuraavaan kohteeseen. Kun eleen paikkaehdot toteutuvat, otetaan talteen määritetyt millisekunnit, jonka aikana käden tulee siirtyä seuraavalle pisteelle, tai tunnistus aloitetaan alusta. Kun tunnistus on käyty lopulta kokonaan läpi, kerrotaan pääohjelmalle, että ele on tunnistettu.



Kuva 19. Ristieleen visualisointi

Kuva 19 havainnollistaa, miten pelaajan on liikutettava kätensä punaisten ympyröiden kohdalle numeroidussa järjestyksessä. Aikaa siirtymiseen on 1500 millisekuntia. Keskimmäinen piste ei ole fyysisesti täysin oikeassa paikassa, mutta käsi kulkee kuitenkin tämän alueen lävitse. Tarkempaan tunnistukseen voisi laskea käsin pisteen alavartalon ja ylävartalon väliltä. Syvyyskuvaa katsottuna oikea käsi on vasemmalla puolella, mutta ohjelmisto peilaa automaattisesti pelaajan luurankomallin, joten piirretty luuranko vastaa näyttöpäätteen edessä seisovaa pelaajaa. [2, s. 24.]

5.4.6 Allekirjoitus

Tuloksia laskettaessa ohjelma tallentaa kovalevylle käsillä ilmaan piirretyn nimen tai kuvion, mikäli pelaaja pääsee ennätyslistalle. Tunnistus tapahtuu seuraamalla käden etäisyyttä suhteessa pelaajan vartaloon. Kun pelaajan käsi on työntynyt tarpeeksi paljon eteenpäin, liikettä seurataan ja kaikki kaksiulotteisessa koordinaatistossa olevat pisteet tallennetaan. Molemmat kädet voivat piirtää erikseen tai yhdessä. Kuvasta 20 käy suurin piirtein ilmi, minkälaiseen tarkkuuteen käden seuranta pystyy.

Kuva 20. Allekirjoitus

Huomioitavaa on kuitenkin se, että ohjelma käyttää alempaa resoluutiota tunnistamiseen kuin mahdollista. Tämä taas johtuu siitä, että kehitystyökaluissa luurankomallin käyttö on rajattu 320 x 240 resoluutiolle, koska työkalut ovat rakennettu konsolilla käytettävien kirjastojen päälle. Kinect on siis täysin kykenevä paljon tarkempaan tunnistamiseen.

5.5 Lopputulos

Esimerkkisovelluksessa päästiin mainiosti asetettuihin tavoitteisiin. Suurimmaksi haasteeksi nousi uuden kielen ja ohjelmointiympäristön opettelu sekä ongelmat ajureiden kanssa, minkä takia Windows 7 ei tunnistanut Kinectin mikrofonia. Tästä johtuen en päässyt koskaan testaamaan puheentunnistusta omalla työkoneella, vaan tarvitsin tähän lainattua kannettavaa. Olin positiivisesti yllättynyt, kuinka hyvin ohjelman arkkitehtuuri toimi, varsinkin testatessani ensimmäistä kertaa puheentunnistusta, jonka olin lisännyt täysin pimennossa siitä, tulisiko se ylipäätänsä toimimaan.

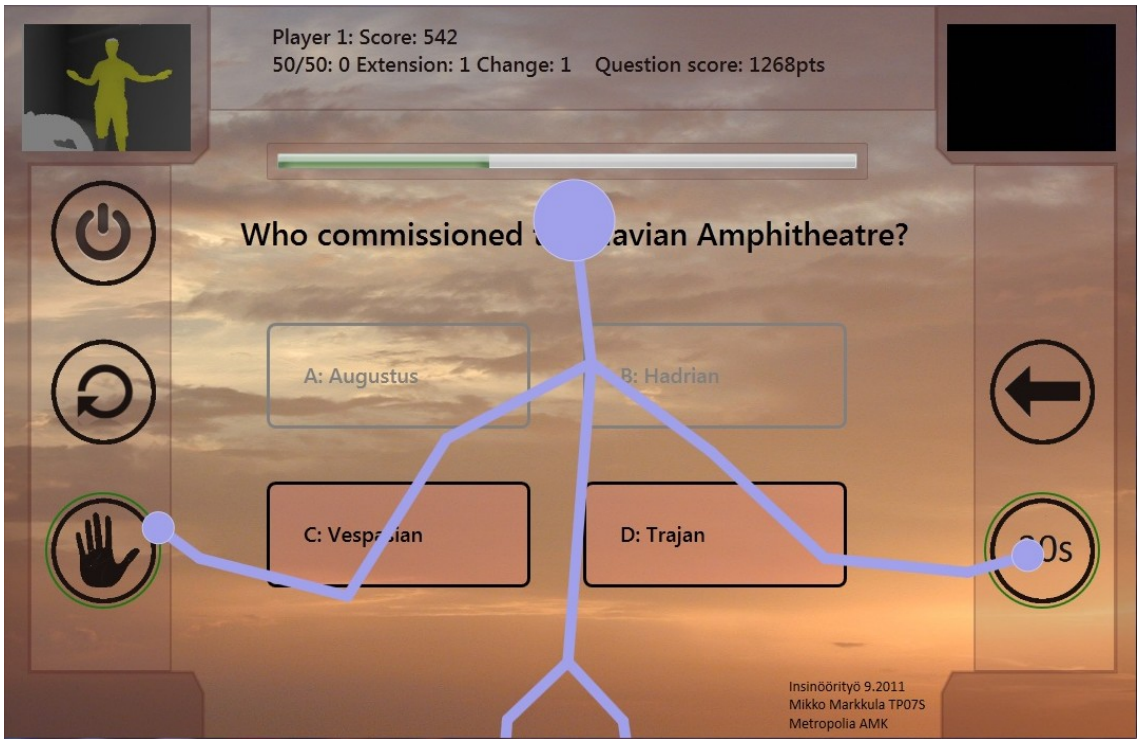
Ominaisuudet, kuten piirrettävä käsikirjoitus, ei ollut alkujaan ollenkaan suunnitelmissa, mutta tajusin ohjelmoidessa, kuinka helppoa se oikeastaan olisi. Olin myös aikaisemmin kirjoittanut samanlaisen ominaisuuden sovellukseen, jota tein päivätyössä. Tästä heräsi myös ajatus siitä, että olisi voinut lisätä sovellukseen yhden ominaisuuden lisää, jossa pelaaja voi kysymysruudulla piirtää ilmaan kirjaimia, jotka ohjelma lukisi käyttäen optista merkinlukua. Aikataulurajoitteiden takia ominaisuus ei päässyt ohjelmaan.

5.5.1 Sovelluksen lopullinen ilme

Kuva 21. Kysymystyypin valinta

Kuvassa 21 on tultu eteenpäin alkuvalikosta ja valittu yksinpeli. Ruudulla näkyvät kysymystyypit eli historia, musiikki, urheilu sekä kuvataide. Ne voidaan valita joko painamalla nappulaa tai sanomalla aihe ääneen. Pelaaja voi käyttää nappuloiden painamiseen molempia käsiä, mutta ohjelma toimii niin, että nappulaa voisi käyttää vaikkapa vasemmalla jalalla pelkästään vaihtamalla valittu piste luurangosta. Vasemmassa laidassa ovat ylätason komennot eli ohjelman sulkeminen, käynnistäminen uudestaan tai käsien piirtämisen vaihtaminen koko luurangon piirtämiseksi. Nämä kuten muut nappulat toimivat puheentunnistuksella, mutta virhetunnistusten eliminoinnin takia niihin kuuluu etuliite.

Yläkulmissa on syvyyskameran ja värikameran tuottama kuva. Syvyyskamera ei anna värillistä kuvaa ulos automaattisesti, vaan se pitää visualisoida ohjelman sisällä. Se kertoo kuitenkin, jos tietty piste 320 x 240 kokoisessa taulussa kuuluu pelaajaksi tunnistetulle kappaleelle. Väritys pitää kuitenkin toteuttaa itse. En kuitenkaan nähnyt ominaisuuden kirjoittamista kokonaan uusiksi tarpeelliseksi, joten nappasin yhdestä kehitystyökalujen mukana tulleesta esimerkistä koko visualisointimetodin mukaan ohjelmaan.



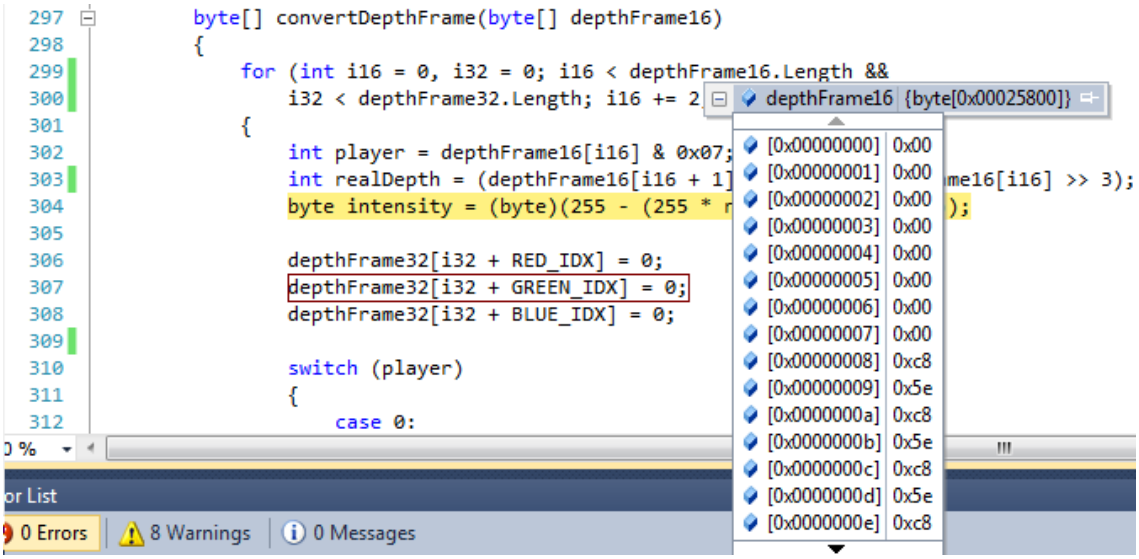
Kuva 22. Kysymysruutu

Kuva 22 on otettu vastaamisruudulta, jossa oikeasta laidassa on kysymyksiin käytettävissä olevat oljenkorret ja vasemmasta laidasta jokaiselle ruudulle yhteiset nappulat. Vakiona pelaajasta näytetään vain kaksi käsi-ikonia mutta tässä tapauksessa koko luuranko piirretään, koska pelaaja on painanut vasemmalta nappulaa, joka valitsee, piirretäänkö joko pelkät kädet vai koko luuranko. Alkujaan ohjelmassa ei pitänyt olla ollenkaan luurangon piirtämistä, mutta huomasin sen helpottavan itse pelaamista.

Vastaamiseen on aikaa 30 sekuntia minkä voi tuplata hyödyntämällä yhden oljenkorsista. Muita oljenkorsia ovat kahden virheellisen vastauksen poistaminen ja kysymyksen vaihtaminen. Jokaista oljenkortta annetaan vakiona käyttöön ainoastaan yksi kappale. Ajan vähentyessä vähenee myös oikeasta vastauksesta saatu pistemäärä, joka taas nousee vaikeustason kasvaessa. Musiikkikysymyksissä soitetaan lyhyt pätkä tunnettua sävellystä, jonka pelaajan pitää tunnistaa. Taidekysymyksissä pitää tunnistaa maalauksia tai veistoksia. Vastatessa väärin tai kysymysten loppuessa pelaajalta pyydetään allekirjoitus, mikäli tulos oli tarpeeksi hyvä ennätyslistalle, jossa näkyy viisi tulosta.

5.5.2 Ohjelmointityö ja ongelmat

Microsoftin virallinen työkalupaketti oli helppo ottaa käyttöön ja C# kielenä oli helppo omaksua, varsinkin kun tunsin Javaa ja C++:tä entuudestaan melko hyvin. Suurempi haaste oli opetella .NET -ympäristön erikoisuuksia, kuten Windows Presentation Foundationin piirtäminen.

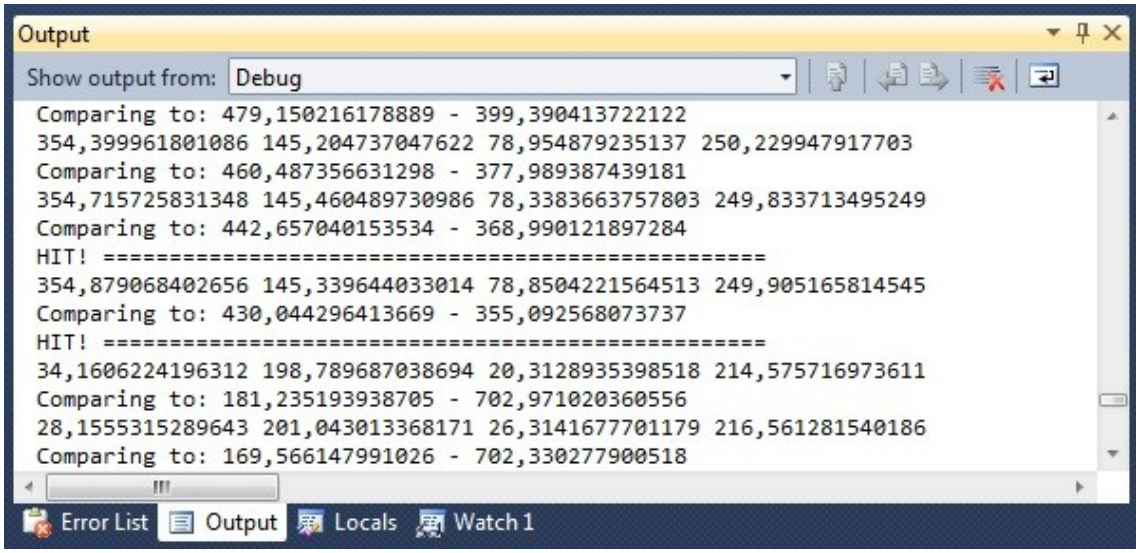


Kuva 23. Visual Studio 2010 debuggeri

Visual Studio 2010 oli alkukankeuden jälkeen erinomainen työkalu ja kuvan 23 testaustoiminnallisuus on paras, mihin olen eri kehitysympäristöjä käyttäessäni törmännyt. Ajonaikaisesti oli mahdollista muunnella muuttujien arvoja mikä helpotti valtavasti nappuloiden ja tekstien sijoittelua, jonka tein täysin käsin hyödyntämättä suunnittelutyökalua.

Itse Kinectin käyttö ohjelmoinnissa oli helppoa. Koska luurankomallin tunnistus löytyy kirjastoista suoraan, ei tarvinnut tehdä muuta kuin käyttää luurangon kolmiulotteisessa avaruudessa olevia pisteitä.

Ohjelma oli täysin mahdollista saada hyytymään huolimattomalla ohjelmoinnilla. Jos muistinvarauksia unohtui nappuloiden tai muiden elementtien piirto-operaatioihin, sillä oli erittäin negatiivinen vaikutus suorituskykyyn. Pääsin lopulta siihen, että ohjelma käytti noin kolmasosan koneeni suoritustehosta. Tehoiltaan kone on aavistuksen nopeampi, kuin työkalupaketin vaatimuksiksi esitetään. [2, s. 4.]



Kuva 24. Perinteistä testaamista konsolissa

Pelaajan eleiden tunnistaminen oli erittäin raskasta, osaltaan siitä syystä, että pelaajapalautteeseen reagointi ei ollut ohjelmassa kaikkien ihanteellisimmassa paikassa, eli siinä tapahtumankäsittelijässä, joka päivittää luurangon pisteitä. Kuvassa 24 eleitä testataan perinteisesti konsolia käyttäen. Päädyin lopulta siihen, että eleiden tunnistus tehdään ainoastaan joka viides sykli. Se tarjosi silti kiitettävän tarkkuuden ja toimi toivotusti. Nappuloiden tunnistustapa aiheutti myös sen ongelman, että liikutettaessa kättä sivusuunnassa, käsi saattaa tahattomasti liikkua eteenpäin tarvittavan matkan ja rekisteröidä osuman. Tästä samaisesta syystä todennäköisesti Xbox 360 -käyttöliittymä ei toimi painamalla vaan pitämällä kättä nappuloiden päällä tietyn aikaa.

Äänen tunnistaminen ei toiminut täysin moitteetta. Kehitystyökalujen käyttämä englanninkielinen akustinen malli ei tarjoa luotettavaa tietoa tunnistetun äänen tunnistustarkkuudesta, joten esimerkiksi tuolin narahtaminen tulkittiin suurella varmuudella ohjelman sulkevaksi komennoksi. Vasta kun varmuustekijän nosti 97 prosenttiin, pääsi eroon virheellisistä tunnistuksista. Tämä taas aiheutti sen, että monet selkeästi puhutut

sanat jäivät tunnistamatta, varsinkin jos puhujan ääntämisessä oli toivomisen varaa. On tietenkin muistettava että työkalut ovat yhä beta-asteella ja melko pienen kehittäjätiimin kädenjälkeä.

Yksi iso ongelma oli se, miten peliä tulisi pelata, sillä sovellus tarvitsi vähintään puolitoista metriä etäisyyttä toimiakseen järkevästi. Tämä etäisyys on jo suuri etäisyys monitorista, joten ruudulla näkyvien tekstien tuli olla suurikokoista, että ne pystyi lukea. Kinect onkin suunniteltu laitteeksi olohuoneeseen, jossa televisioruutujen koot ovat huomattavasti monitoreita suuremmat.

6 Yhteenveto

Kinect on hintaansa nähden monipuolinen ja hauska laite varsinkin kokeilunhaluisille harrastajille. Mikään muu laite ei tarjoa täyttä luurankomallinnusta ja muita ominaisuuksia samaan hintaan. Laite on kuitenkin kehitetty olohuoneeseen sijoitettavaksi pelilaitteeksi, eikä sitä ole suunniteltu käytettäväksi tavallisten pöytäkoneiden yhteydessä. Tämä on ilmeistä luurankomallia käytettäessä, kun koko vartalon tunnistamiseen tarvitaan huomattava etäisyys. Avoimen lähdekoodin ajureilla toteutetuissa harrastelijaprojekteissa on esimerkkejä siitä, miten laitetta voi käyttää pelkästään käsien tunnistamiseen, eli Kinectillä voi tehdä paljon muutakin. Tätä ominaisuutta ei kuitenkaan löydy vielä vakiona beta-asteen kehitystyökaluista, mutta sitä on järkevää odottaa, kun Kinect saa virallisen PC-tuen.

Testatessa Kinectiä Xbox 360 -pelikonsolilla luurankotunnistuksen kanssa tuli ongelmia heti, kun pelitila ei ollut täysin ihanteellinen. Tunnistus pätki ja laitteen käyttö ohjaimena oli täysin mahdotonta, kun huoneeseen heijastui kirkasta auringonvaloa. Taustalla oli myös heijastavia materiaaleja, mikä aiheuttaa myös ongelmia syvyyskuvan kanssa. Puheentunnistus tukee ainoastaan englantia ja toimii vasta, kun konsolin asetuksista vaihdetaan sijaintia, mutta se tunnistaa mainiosti jopa huonosti äännettyä englantia. Peleihin en investoinut, joten en päässyt testaamaan paremmiksi luokiteltuja pelejä, ainoastaan ohjaimen mukana tullutta Kinect Adventures -peliä, joka oli hauska hetken ja vetoaa varmasti nuorempiin pelaajiin. Kinectin latenssiongelma on kuitenkin ilmeinen

pelejä pelatessa, mutta se ei haitannut tämänkaltaisessa kevyessä pelaamisessa ollenkaan.

Isoin este Kinectin läpilyönnille tavallisissa koneissa on se, että sensoria ei ole suunniteltu käytettäväksi pöytäkoneen yhteydessä. Luurankotunnistuksen vaatima etäisyys on huomattava ja syvyyskuvan virheetön käsittely vaatii liki metrin etäisyyden. Sovelluskehitystä hankaloittaa juuri erilaiset ympäristöt ja etäisyydet joissa laite joutuu toimimaan. Harrastelijoiden projekteissa on kuitenkin esimerkkejä siitä, miten sensoria voi käyttää täysin eri tavalla kuin konsolille suunniteltuna. Laitekanta on myös melko pieni, joten yhdenkään yrityksen ei ole järkevää luoda kaupallista sovellusta PC:lle. Kaupallisiin tarkoituksiin kehittäminen on kuitenkin mahdollista avoimen lähdekoodin ajureilla. Virallisen PC-julkaisun myötä ja oletettavasti hinnan pudotessa voi odottaa kaupallisia Kinect-sovelluksia, mutta tämä vaatii myös digitaalisen jakelualustan, kuten Steamin, koska jo konsolilla laitteelle on riskialtista luoda suuren budjetin pelejä.

Kinectin käyttö sovelluskehityksessä oli erittäin vaivatonta, eikä vaatinut monimutkaista matemaattista osaamista. Varsinkin puheentunnistuksen hyödyntäminen oli yllättävän helppoa, ja sovellusta oli täysin mahdollista käyttää pelkin äänikomennoin. Käytännössä annettujen työkalujen avulla tarvitsee ainoastaan seurata yhtä tai useampaa pistettä kolmiulotteisessa avaruudessa. Esimerkkien pohjalta puheentunnistus oli niinkin yksinkertaista kuin omien sanojen keksimistä ja linkittämistä haluttuihin toimintoihin. Sovellusta kehittäessä on olennaista muistaa etäisyydet, joilta laitetta voi käyttää. Vaikka käsien tunnistaminen on mahdollista, vietäessä tunnistettava kappale liian lähelle sensoria syvyyskuvassa ilmenee virheitä.

Työ oli mielenkiintoinen ja sitä tehdessä tuli opittua valtava määrä uutta asiaa, uudesta ohjelmointikielestä aina siihen, miten ohjelma tulee rakentaa. Luotettavien lähteiden löytäminen näin tuoreesta aiheesta oli hankalaa, joten oli pakko käydä läpi lukuisia artikkeleita usein samoista aiheista ja muodostaa niistä jonkinlainen kuva kokonaisuudesta. Työ tehtiin täysin omalla ajalla oma-aloitteisesti päivätyön ohella, jossa käydään läpi täysin erilaisia aiheita eri ohjelmointikielellä. Suurin haaste muodostuikin siitä, miten löytää aika ja motivaatio ei-palkitsevaan tutkimustyöhön ja ohjelmointiin uudella kielellä uudella ohjelmointityökalulla.

Lähteet

1 Wii. 2011. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Wii>. Luettu 5.10.2011.

2 Konsoleiden ja pelien myyntitilastoja. 2011. Verkkodokumentti. <http://www.vgchartz.com/>. Luettu 10.10.2011.

3 Playstation 3. 2011. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/PlayStation_3>. Luettu 5.10.2011.

4 Kinect. 2011. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Kinect>. Luettu 25.10.2011.

5 Kinect hacked days after release. 2010. Verkkodokumentti. BBC. <http://www.bbc.co.uk/news/technology-11742236>. Luettu 23.10.2011.

6 Kinect for Windows SDK Beta -kotisivut. 2011. Verkkodokumentti. Microsoft. <http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/default.aspx>. Luettu 20.10.2011.

7 Plunkett, Luke. 2010. The New 360 "Hub" Looks Better Than The Dashboard. Verkkodokumentti. Kotaku. <http://kotaku.com/5563467/the-new-360-dashboard-looks-better-than-the-old-one>. Luettu 25.10.2011.

8 Rajala, Arttu. 2010. Kinect Adventures. Verkkodokumentti. Gamereactor. <http://www.gamereactor.fi/arviot/78594/Kinect+Adventures/>. Luettu 25.10.2011.

9 Gibson, Ellie. 2010. Dance Central Review. Verkkodokumentti. Eurogamer. <http://www.eurogamer.net/articles/2010-11-04-dance-central-review>. Luettu 25.10.2011.

10 Dance Central. 2011. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Dance_Central>. Luettu 22.10.2011.

11 Yin-Poole, Wesley. 2010. Source: MS quadrupling Kinect accuracy. Verkkodokumentti. Eurogamer. <http://www.eurogamer.net/articles/2010-12-17-source-ms-quadrupling-kinect-accuracy>. Luettu 24.10.2011.

12 Hruska, Joel. 2010. Microsoft: Kinect CPU Usage Down to Single Digits. Verkkodokumentti. Hothardware. <http://hothardware.com/News/Microsoft-Kinect-CPU-Usage-Down-to-Single-Digits-/>. Luettu 12.10.2011.

13 Microsoft E3 2011 Press Conference. 2011. Verkkodokumentti. Gametrailers. <http://www.gametrailers.com/video/e3-2011-microsoft/715731>. Luettu 20.9.2011.

14 Schramm, Mike. 2010. Kinect: The company behind the tech explains how it works. Verkkodokumentti. Joystiq. <http://www.joystiq.com/2010/06/19/kinect-how-it-works-from-the-company-behind-the-tech/>. Luettu 24.10.2011.

15 Microsoft Kinect Teardown. 2011. Verkkodokumentti. Ifixit. <http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/2>. Luettu 21.10.2011.

16 THE PS1080. 2011. Verkkodokumentti. PrimeSense. <http://www.primesense.com/en/technology/114-the-ps1080>. Luettu 21.10.2011.

17 Programming Guide for Kinect SDK Beta - Beta 1 Draft Version 1.1 – July 22, 2011. Verkkodokumentti. Microsoft. <http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/docs/programmingguide_kinectsdk.pdf>. Luettu 25.10.2011.

18 Reetz, Daniel; Kariluoma, Matti. 2010. Kinect Hacking 103: Looking at Kinect IR Patterns. Verkkodokumentti. Futurepicture. <http://www.futurepicture.org/?p=116>. Luettu 25.10.2011.

19 Beamforming. 2011. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/Beamforming>. Luettu 24.10.2011.

20 Kinect – Why You Should Get It. 2011. Verkkodokumentti. Video Game Legion. <http://videogamelegion.blogspot.com/2011/02/kinect-why-you-should-get-it.html>. Luettu 25.10.2011.

21 Carmody, Tim. 2010. How Motion Detection Works in Xbox Kinect. Verkkodokumentti. Gizmodo. <http://gizmodo.com/5681078/how-motion-detection-works-in-xbox-kinect>. Luettu 24.10.2011.

22 Bierton, David. 2011. Tech Report: Kinect - The Latency Question. Verkkodokumentti. IQGamer. <http://imagequalitymatters.blogspot.com/2010/08/tech-report-kinect-latency-question.html>. Luettu 24.10.2011.

23 Sinclair, Brendan. 2011. Kinect has problems recognizing dark-skinned users? Verkkodokumentti. Gamespot. <http://www.gamespot.com/news/6283514/kinect-has-problems-recognizing-dark-skinned-users>. Luettu 24.10.2011.

24 EyeToy. 2011. Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/EyeToy>. Luettu 22.10.2011.

25 Ferro, Mike. 2010. Can the PS2 EyeToy and PS Eye compare to the Kinect? Verkkodokumentti. Gamer.Blorge. <http://gamer.blorge.com/2010/08/01/can-the-ps2-eyetoy-and-ps-eye-compare-to-the-kinect/>. Luettu 22.10.2011.

26 Kinect Hacking. 2011. Verkkodokumentti. <http://www.ladyada.net/learn/diykinect/>. Luettu 23.10.2011.

27 Catone, Josh. 2011. Microsoft Now "Excited" by Kinect Hacks. Verkkodokumentti. Mashable Tech. <http://mashable.com/2010/11/20/microsoft-kinect-hacks/>. Luettu 25.10.2011

28 Kokoelma harrastelijaprojekteja. 2011. Verkkodokumentti. <http://www.kinecthacks.com/>. Luettu 25.10.2011.

29 Kinect Minority Report. 2010. Verkkodokumentti. Freenect – Kinect projects. <http://www.freenect.com/kinect-minority-report>. Luettu 25.10.2011.

30 Bouffard, Patrick. Quadrotor Kinect. 2011. Verkkodokumentti. Youtube. <http://www.youtube.com/watch?v=eWmVrfjDCyw>. Luettu 25.10.2011

31 Avoimen lähdekoodin ajureiden kotisivut. 2011. Verkkodokumentti. <http://openkinect.org/>. Luettu 24.10.2011.

32 OpenNI kotisivut. 2011. Verkkodokumentti. <http://openni.org/>. Luettu 24.10.2011.

33 Hinchman, Will. 2011. Kinect for Windows SDK beta vs. OpenNI.

Verkkodokumentti. Vectorform. <http://labs.vectorform.com/2011/06/windows-kinect-sdk-vs-openni-2/>. Luettu 26.10.2011.

34 Fernandez, Dan. 2011. Kinect for Windows SDK Quickstarts. Verkkodokumentti. Microsoft. <http://channel9.msdn.com/Series/KinectSDKQuickstarts>. Luettu 13.10.2011.

35 Windows Presentation Foundation. 2010. Verkkodokumentti. Microsoft. <http://msdn.microsoft.com/en-us/library/ms754130.aspx>. Luettu 21.10.2011.

36 An open source ADO.NET provider for the SQLite database engine. 2010. Verkkodokumentti. <http://sqlite.phxsoftware.com/>. Luettu 3.10.2011.

37 SQLiten kotsivut. 2011. Verkkodokumentti. <http://www.sqlite.org/>. Luettu 3.10.2011.