Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Dušan Peternelj

Uporaba naprave Kinect

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Borut Batagelj

Ljubljana 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00312/2012
Datum: 16.05.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **DUŠAN PETERNELJ**

Naslov: **UPORABA NAPRAVE KINECT**
USAGE OF THE DEVICE KINECT

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V svoji diplomski nalogi podrobneje predstavite Microsoftovo igralno napravo XBOX Kinect, ki se veliko uporablja tudi v kombinaciji z računalnikom in s tem nadomešča standardne vmesnike za interakcijo človeka z računalnikom.

Predstavite tudi sorodne brezdotične vmesnike ter osnovo za začetek dela s samo napravo Kinect. V ta namen izdelate tudi preprosto aplikacijo, ki bo izkoriščala glavne senzorje naprave.

Mentor:

viš. pred. dr. Borut Batagelj

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Dušan Peternelj, z vpisno številko **63060331**, sem avtor diplomskega dela z naslovom:

Uporaba naprava Kinect

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Boruta Batagelja
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 5. julija 2012 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju viš. pred. dr. Borutu Batagelju za potrpežljivost, nasvete in podporo pri izdelavi diplomske naloge.
Iskrena hvala tudi staršem, ki so mi omogočili študij v Ljubljani in me podpirali vsa leta študija. Hvala tudi drugim, ki so mi pomagali, da sem uspešno zaključil študij.

Kazalo

Terminologija

Senzor Microsoft Kinect – igralni pripomoček za zajem slike in senzor gibanja. V nadaljevanju bo uporabljena okrajšava Kinect.

Microsoft Kinect SDK – zbirka orodij, primerov, specifikacij za delo z Microsoft Kinectom.

IR – Infrared Ligt – infrardeča svetloba.

FPS - Frames Per Second – število okvirjev ali slik na sekundo – koliko slik naredi v sekundi, izraženo tudi v Hz.

Piksel – Pictorial element, Pixel – najmanjša naslovljiva enota slike.

Bayer filter – barvno filtrirno polje.

NUI – Natural User Interface – naravni uporabniški vmesnik.

CMOS – Complementary Metal Oxide Semiconductor – komplementarni kovinsko-oksidni polprevodnik.

RGB (Red, Green, Blue) – barvni model sestavljen iz rdeča,zelene in modre barve.

YUV – barvni model, pri katerem se komponente barvnega sistema modela RGB ločijo na svetlost (Y) in barvo (UV).

3D – tridimenzionalni.

Port – vrata.

PS – Igralna konzola PlayStation.

SDK - Software Development Kit - orodja za razvoj aplikacij.

PC - Personal Computer – osebni računalnik.

API – Application Programming Interface – vmesnik uporabniškega programa.

Runtime, Runtime enviroment – izvajalnik kode, izvajalno okolje.

Povzetek

Cilj diplomske naloge je predstavitev senzorja Microsoft Kinect in opis izdelave preproste aplikacije, ki omogoča interakcijo človeka z računalnikom s pomočjo Kinecta.

V prvem delu diplomske naloge je opisana naprava Kinect, njena predhodnica in sorodne naprave. Večina teh naprav omogoča interakcijo z igralnimi konzolami oziroma igrami na bolj interaktiven način – več gibanja. Podana sta tudi dva vidika uporabnosti Kinecta. Narejena je primerjava sorodnih naprav s Kinectom. Predstavljeni so najbolj aktualni SDK, knjižnice in gonilniki, ki jih lahko uporabljamo s Kinectom.

V drugem delu je podana kratka postavitev delovnega okolja in predstavljen razvoj preproste aplikacije za interakcijo človeka s Kinectom in računalnikom. Pri tem se bo uporabljalo programsko okolje C# v povezavi z Microsoft Kinect SDK različice 1.5.

Ključne besede: Senzor Kinect, NUI – naravni grafični vmesnik, Bayernov filter, RGB kamera, projektor IR, IR senzor, upravljalnik Wii, PlayStation Eye, PlayStation Move, Asus Xtion Pro, Asus Xtion Pro Live, OpenNi – Nite, OpenKinect – libfreenect, CL NUI Platform, VRUI SDK, Microsoft Kinect SDK, upravljanje aplikacij z gibi.

Abstract

The main goal of this diploma thesis is to present the Microsoft Kinect sensor and to describe the making of a simple application that allows human interaction with the computer through the help of Kinect.

The first part of the diploma thesis describes the device Kinect, its predecessor and its related devices. Most of these devices allow us to interact with game consoles and games in a more interactive way - more movement with the body. Also two aspects of Kinect use are given. A comparison of the related devices with Kinect is made. The most current SDKs, libraries and drivers that can be used with Kinect are also presented.

In the second part the installation of the working environment and the development of a simple application for human interaction with Kinect and the computer are shown. While doing so the C# programming environment in connection with Microsoft Kinect SDK version 1.5 will be used.

Keywords: Microsoft Kinect, NUI – Natural user interface,
Bayer filter, RGB camera, IR projector, IR sensor, Wii Remote,
PlayStation Eye, PlayStation Move, Asus Xtion Pro, Asus Xtion Pro Live,
OpenNi – Nite, OpenKinect – libfreenect, CL NUI Platform, VRUI SDK,
Microsoft Kinect SDK, control applications with gestures.

Poglavje 1

Uvod

Kinect[17] je igralni pripomoček za konzolo XBOX 360 (Slika 1.1). Na konzolo je priključen preko USB-ja oz. posebnega priključka. Uporablja se za upravljanje igre s sledenjem igralčevih gibov in za glasovno vodenje igre. Ima svojo logiko za procesiranje okolja, gibov in zvoka. Na trgu je dobre 2 leti in ponuja številne možnosti uporabe v robotiki, medicini itd. Zaradi priklopa preko vrat USB je toliko bolj zanimiv tudi za raziskovalce, ki ga uporabljajo za različne projekte. V času obstoja naprave se je pojavilo več različnih primerov uporabe Kinecta. S februarjem 2012 je izšel tudi Kinect, ki je namenjen zgolj za uporabo z računalnikom. V poglavju 2.9. je navedenih nekaj aktualnih projektov, ki uporabljajo Kinect.

Slika 1.1: Senzor Kinect.

Za pisanje diplomske naloge na temo Kinecta smo se odločili predvsem zato, da bi predstavili Kinect, ter način programiranja le tega s pomočjo programskega jezika C#. V prvem delu diplomske naloge bomo predstavili napravo Kinect in njeno zgradbo. Opisali bomo tudi sorodne naprave in naredili primerjavo naprav s Kinectom. Na koncu prvega dela bomo navedli tudi možne SDK, knjižnice in gonilnike, ki so primerni za Kinect. Navedli bomo tudi nekaj aktualnih projektov povezanih s Kinectkom

V drugem delu bomo s pomočjo programskega okolja C# in Microsoft Kinect SDK različice 1.5 razvili preprosto aplikacijo, ki bo izkoriščala napravo Kinect oziroma njene senzorje.

Poglavje 2

Kinect

Na začetku poglavja bomo na kratko predstavili napravo, ki je bila predhodnik Kinecta, zgodovino razvoja Kinecta in princip delovanja le tega. V nadaljevanju bomo opisali posamezne komponente Kinecta in navedli dva vidika uporabnosti. Nato sledi opis sorodnih naprav in primerjava le teh s Kinectom. Proti koncu poglavja bomo navedli možne SDK-je, knjižnice in gonilniki, ki jih lahko uporabljamo s Kinectom. Na koncu poglavja bomo predstavili nekaj zanimivih projektov, ki so bili izvedeni s pomočjo Kinecta.

2.1 Predhodnik Kinecta

Predhodnik Kinecta je XBOX Live Vision (Slika 2.1). Uporabljal se je kot kamera za zajem slike in spletne pogovore z raznimi namenskimi aplikacijami. Ni imel vgrajenega mikrofona.

Ima možnost dveh resolucij. Prva resolucija je 640 x 480 slikovnih točk, druga pa 320 x 240 slikovnih točk. Zajem videa je potekal s hitrostjo 30 slik/sekundo oz. 30 Hz, pri manjši resoluciji pa je zajem 60 slik/sekundo oz. 60 Hz. Slike smo lahko zajemali z ločljivostjo 1,3 megapikslov.
Z namenskim programom je bilo možno nastavljati naslednje učinke:

- vodnega.
- ostrega.
- točkastega.

Slika 2.1: Live Vision.

2.2 Zgodovina razvoja Kinecta

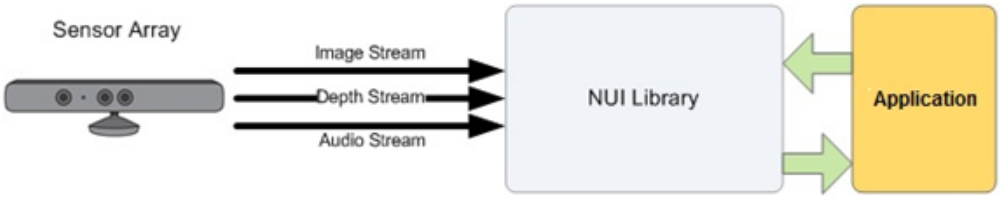
Kinect je bil razvit na podlagi programske opreme iz podjetja Rare, ki je hčerinsko podjetje podjetja Microsoft Game Studios. Izraelsko podjetje PrimeSense je prispevalo njihovo tehnologijo, imenovano intervalna kamera (angl. range camera), ki z infrardečim projektorjem, kamero in namenskim procesorjem skrbi za spremljanje gibov v 3D prostoru. To jim omogoča tehnologija imenovana svetlobno kodiranje (angl. light coding), ki posnete slike sestavlja v 3D sliko.

2.3 Princip delovanja Kinecta

NUI je kratica za naravni uporabniški vmesnik. To je skupek tehnologij, ki omogoča, da stvari upravljamo brez upravljalnikov. Edini naravni upravljalnik je naše telo. S tem vmesnikom lahko postavimo okolje, ki se interaktivno prilagaja uporabniku in ga prepoznava po zgradbi telesa, glasu, obraznih značilnostih. Bistvo tehnologije je, da preko senzorjev, kamer in mikrofonov zaznava gibe telesa oz. njegovih delov v realnem času. Na ta zajem podatkov se mora tudi ustrezno odzvati v realnem času. To pa predvsem sloni na knjižnicah za obdelavo vhodnih podatkov Imamo senzor, ki preko NUI zaznava in knjižnici NUI pošilja naslednje tri toke podatkov:

- slikovni tok podatkov,
- globinski tok podatkov,
- zvočni tok podatkov.

Knjižnica NUI skrbi, da neobdelane podatke prilagodi za naše aplikacije. Aplikacije komunicirajo s knjižnico NUI po potrebi, odvisno od ukazov, ki jih izvajajo naše aplikacije. Zajemanje in procesiranje podatkov poteka v realnem času. Shema zajema podatkov (Slika 2.2) nam prikazuje potek zajema od naprave do naše aplikacije.

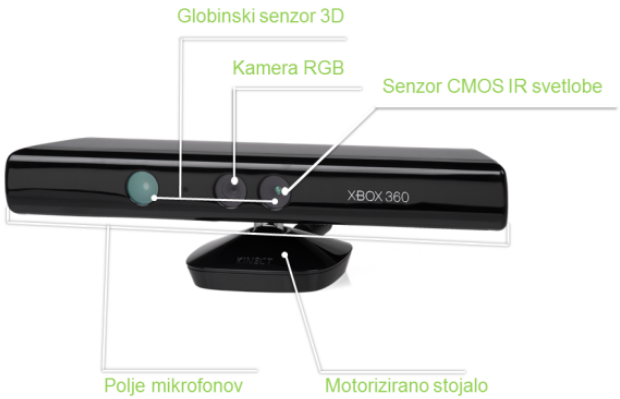


Slika 2.2: Shema zajema podatkov.

2.4 Komponente Kinecta

Kinectovo ohišje ima obliko podolgovatega droga, ki je pritrjen na majhno stojalo. Notranjost ohišja je skrbno oblikovana. V notranjosti je ventilator, ki skrbi za pretok zraka po napravi in s tem onemogoča pregrevanje senzorjev. Ventilator krmili logika in se vklopi ob določeni temperaturi – okrog 70 °C. Kinect je sestavljen iz naslednjih delov [10] (Slika 2.3) :

- kamere RGB,
- senzorja 3D globine, sestavljenega iz:
  - laserskega projektorja IR,
  - senzorja IR – črno-beli senzor CMOS,
- polja mikrofonov – 3D zvok,
- senzorja naklona in pospeška,
- stojala z motorčkom.



Slika 2.3: Deli senzorja Kinect.

Območje zajema slike z napravo in Xbox programsko opremo je med 1,2 in 3,5 m; področje, v katerem zaznava objekte, pa 6 m2. Če slike zajemamo z lastnimi programi, se lahko območje giblje od 0,5 do 6 m ob pravi svetlobi. Horizontalni kot zajema je 57°, vertikalni kot 42° in diagonalni 70°.

Na Kinectu (Slika 2.4) brez ohišja so lepo razvidni posamezni deli [21, 59].

Slika 2.4: Sprednja stran brez ohišja.

Posamezni deli Kinecta so podrobneje opisani v nadaljevanju. Kinect je zelo občutljiv na močno dnevno svetlobo, zato je tudi namenjen notranji uporabi.

2.4.1 RGB kamera

Kamera uporablja čip MT9M112. Vsebuje 1.3 megapiksel SOC CMOS Digital Image senzor s 1280 x 1024 aktivnimi slikovnimi točkami. Uporablja optični format 5:4 in Bayernov RGB filter.

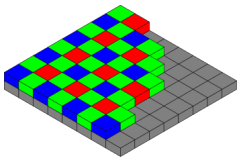
Predhodnik čipa MT9M112 je bil čip z oznako MT9v112. Vseboval je SOC VGA CMOS DIGITAL IMAGE senzor s 640 x 480 aktivnimi slikovnimi točkami. Uporablja optični format 4:3 in Bayernov RGB filter. Uporabljal se je v prvih različicah kamere.

Kamera ima privzeti podatkovni zajem barvne slike. Zajem poteka v resoluciji 640 x 480 slikovnih točk s hitrostjo 30 Hz. Zajema lahko tudi z resolucijo 1280 x 1024 slikovnih točk. Pri tem načinu pa hitrost zajema pade na 12 do 15 Hz.

Uporablja se Bayernov barvni filter [5]. To je polje barvnih filtrov, postavljenih na kvadratno mrežo svetlobnih senzorjev. Filter se uporablja v večini digitalnih enoprocesorskih svetlobnih senzorjih. Ti pa se nahajajo v večini sodobnih kamer, videokamer in optičnih čitalnikov.

Osnovna enota (Slika 2.5a) pri Bayernovem filtru je sestavljena iz štirih celic. Dve celici sta pokriti z zelenim, ena celica z rdečim in ena celica z modrim filtrom. S pomočjo interpolacije se nato dobi barvno sliko – RGB vzorec. Polje (Slika 2.5b) je sestavljeno iz več Bayernovih enot.

(a) Bayernova enota.



(b) Bayernovo polje.

Slika 2.5: Sestavna dela Bayernovega filtra.

Bayernov filter nam vrača neobdelane podatke z resolucijo 1280 x 1024 slikovnih točk. Procesna logika jih nato stisne in pretvori v RGB obliko [32]. Nato jih posreduje izvajalniku kode, ki jih mora raztegniti in posredovati naši aplikaciji. S tem ko stisnemo podatke, omogočimo prenos podatkov

s hitrostjo 30 Hz (slik/okvirjev na sekundo). Uporabljata se dva barvna formata:

- **RGB** – 32-bitni, linearni X8R8G8B8 – formatirana bitna slika X8R8G8B8 – je barvni format, ki se uporablja pri RGB obliki in ima za prikaz vsake barve namenjenih 8 bitov;
- **YUV** – 16-bitni in 15 Hz – zaseda manj prostora in medpomnilnika – neobdelani podatki [22].

Slika 2.6 je primer slike, posnete s Kinectovo RGB kamero.

Slika 2.6: Slika, pridobljena iz RGB kamere.

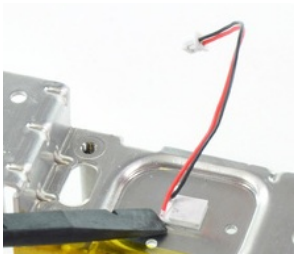
2.4.2 IR senzor globine

Je sestavljen iz dveh delov – projektorja IR svetlobe in senzorja IR svetlobe. Povezana sta v stereo način [33]. To pomeni, da simulirata način daljnogleda [6] oz. človeškega vida, ki se uporablja za izdelavo 3D slik. Ta proces se imenuje stereo fotografiranje. Pri tem procesu gre za postavitev projektorja in kamere pod določenim kotom. S pomočjo triangulacije se izračuna oddaljenost osebe oz. objekta od senzorja. S tem pravilno in lažje umeščamo objekte v 3D prostor.

2.4.2.1 Projektor IR svetlobe

Slika 2.7: Projektor IR svetlobe.

Projektor IR svetlobe (Slika 2.7) uporablja 830 nm lasersko diodo. Izhodna moč projektorja je okoli 60 mW. Za hlajenje projektorja skrbi Peltierjev element (Slika 2.8), ki leži med projektorjem in aluminijskim držalom.



Slika 2.8: Peltierjev element.

Laserski projektor IR svetlobe, ki se nahaja na sprednji levi strani Kinecta, uporablja tehnologijo strukturirane svetlobe [34]. Strukturirana svetloba pomeni, da ima svetloba oz. projicirana svetloba nek vzorec – delno naključen vzorec. S tem vzorcem pa si pomaga pri izračunih oddaljenosti točk od naprave. Pri izračunu upošteva tudi sosedne točke in njihovo lego.

Kinectov laserski projektor s pomočjo leče razprši infrardečo svetlobo v obliki vzorca [60] (Slika 2.9). Vzorec je sestavljen iz 3 x 3 velike mreže, kar

Slika 2.9: Primer IR vzorca.

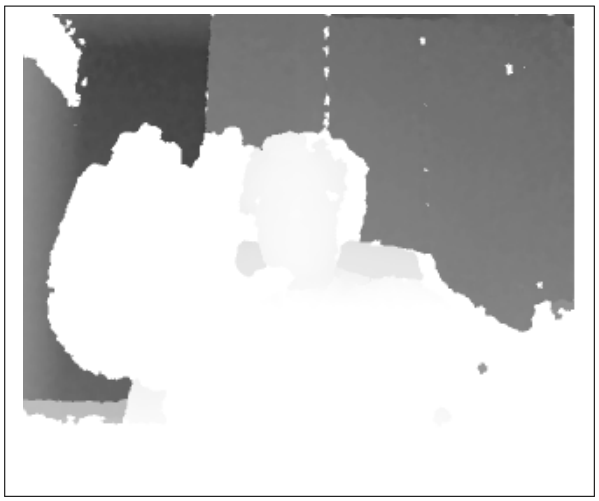
nanese devet polj. Eno polje v mreži vsebuje 211 x 165 vzorčnih pik. V vsakem polju se nahaja točka, ki je svetlejša od ostalih [20].

2.4.2.2 Senzor IR svetlobe

Senzor je sestavljen iz čipa MT9M001C12STM. To je Megapixel CMOS Digital Image senzor s 1280 x 1024 aktivnimi slikovnimi točkami [1]. Uporablja optični format 5:4. Zajem slike s senzorjem CMOS je črno-beli (monokromatski) oz. enobarvni v odtenkih ene barve (Slika 2.10). Natančnost zajema ima 11 bitov, to pomeni 2048 različnih vrednosti (211 = 2048 stopenj natančnosti pri merjenju globine).

Zajema sliko z naslednjimi možnimi resolucijami:

- 640 x 480 slikovnih točk,
- 320 x 240 slikovnih točk,
- 80 x 66 slikovnih točk.



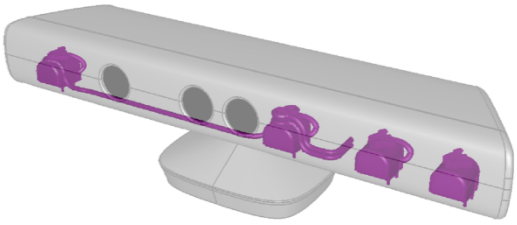
Slika 2.10: Slika, pridobljena s CMOS senzorjem.

Hitrost zajema slike je 30 slik/sekundo ali 30 Hz.

2.4.3 Polje mikrofonov

Sestavljajo ga štirje mikrofoni v obliki kapsul (Slika 2.11). Vsak kanal se procesira s 16 biti in frekvenco vzorčenja 16 kHz. Senzor ima s strojno integracijo urejene naslednje filtre:

- večkanalno izničevanje odmeva,
- zmanjševanje in izločanje šumov okolice.



Slika 2.11: Kinect mikrofoni.

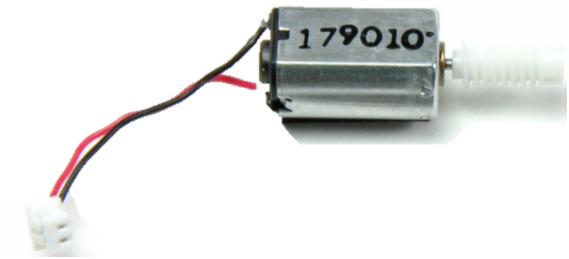
2.4.4 Senzor naklona in pospeška

Za senzor se uporablja čip Kionix MEMS KXSD9. Z njim se ugotavlja položaj oz. naklon Kinecta in tudi njegov pospešek.

2.4.5 Stojalo z motorčkom

V stojalu je vgrajeni motorček (Slika 2.12). Kinect je nanj povezan z vzvodom. Motorček skrbi, da se Kinect nagiba za $\pm 27°$.

Priporočeno je, da se motorček ne zažene več kot 15-krat v 20 sekundah, ker pride do hitre obrabe ležajev. Za napajanje porabi 5 V.



Slika 2.12: Motorček Kinecta.

2.5 Vidika uporabe

2.5.1 Igralni

Osnovni namen Kinecta je pripomoček za igranje iger na konzoli XBOX 360. S Kinectom je igranje iger bolj živo. V igri sodeluješ z vsem telesom. Ni treba več sedeti in držati igralnega upravljalnika. Igre igraš z gibi telesa. Pri igranju se razgibaš, kar je zdravju prijazneje kot pa sedenje po nekaj ur na istem mestu. Opušča splošni namen igralnih upravljalnikov.

2.5.2 Tehnološki/raziskovalni

Kinect je dokaj cenovno dosegljiva naprava, ki v svoji zgradbi ponuja globinski senzor, kamero RGB, polje mikrofonov.Logika v njem sledi telesu in njegovim delom, locira lokacijo zvoka, uporablja različne filtre zvoka.

To so predvsem izkoristila razna podjetja, raziskovalne ustanove in univerze za svoje projekte. Ponujajo različne možnosti adaptacije Kinecta za različne namene.

Predvsem so to projekti v medicini, robotiki, plesu, športu – fitnes, interaktivnem izobraževanjem itd. S prihodom komercialne različice Kinecta za PC so se te možnosti uporabe še bolj razširile.

2.6 Sorodne naprave

Sorodnih igralnih pripomočkov, ki spodbujajo igralca, da se začne gibati ob igranju iger, je iz leta v leto več. Trenutno najbolj aktualni so naslednji:

- naprave Nintendo:
  - daljinski upravljalnik Wii,
  - Wii Balance,
- naprave Sony:
  - PS Move,
  - PS Eye,
- naprave Asus:
  - Xtion Pro,
  - Xtion Pro Live.

2.6.1 Naprave Nintendo

2.6.1.1 Daljinski upravljalnik Wii

Slika 2.13: Wii upravljalnik [36].

Je priložen h konzoli Nintendo Wii kot primarni upravljalnik (Slika 2.13). Sestavljajo ga naslednji deli:

- spomin velikosti 16 KB EEPROM (16,3 kilobajta),
- zvočnik,
- 3-smerni senzor gibanja oz. pospeška ADXL330,
- optični senzor PixArt,
- 8 digitalnih gumbom in digitalni smernik (D-pad),
- vmesnik Bluetooth,
- naprava za tresenje,
- posebna priklopna vrata (angl. port) za dodatke.

Posebni dodatki zanj so:

- Nunchuk,
- Wii Zapper,
- Wii Wheel,
- Classic Controler,
- Classic Controler Pro.

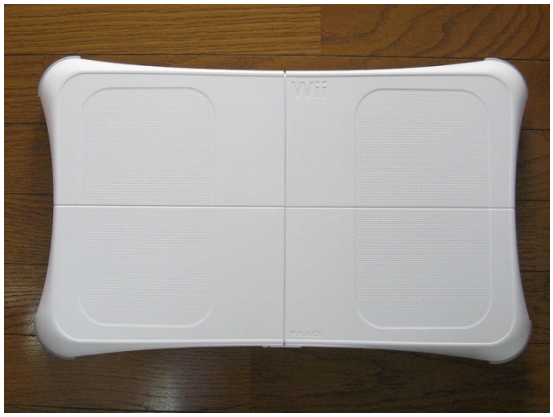
Upravljalniku je priložena naprava Senzor Bar (Slika 2.14), ki se poveže na Wii konzolo. Senzor bar ima tanko podolgovato obliko. Na njem je skupno 10 IR LED lučk, na robovih naprave po 5. Te lučke skrbijo, da s pomočjo triangulacije v vidnem polju naprave najde upravljalnike. Najde jih preko optičnega senzorja PixArt, ki se nahaja v upravljalnikih.

Upravljalnike zaznava do razdalje 10 metrov. Za operativno upravljanje iger in naprave se razdalja prepolovi na 5 metrov.

Slika 2.14: Naprava Sensor bar.

2.6.1.2 Wii Balance Board – deska za ravnovesje Wii

Je dodatek za Wii konzolo. Povezuje se preko Bluetooth povezave. Oblikovno je podobna tehtnici (Slika 2.15). Na njej so 4 polja. V vsakem polju so senzorji pritiska. Uporabljajo se za merjenje gravitacijske sredine uporabnika in njegove teže. Težo izračuna bolj natančno kot navadna tehnica. Največja obremenitev, ki jo zdrži, je 300 kg. Uporablja se predvsem pri raznih športnih igrah, npr. deskanju, kjer je vse odvisno od nagiba uporabnika. Primerna je tudi za zdravstvene namene, ker lahko ob določenih aplikacijah skrbi za pravilno držo uporabnika.



Slika 2.15: Primer naprave Balance Board [37].

2.6.2 Naprave Sony

2.6.2.1 Konzola PlayStation 3 – PS3

Konzola [25] (Slika 2.16) je glavna procesna enota, ki skrbi za delovanje naprav, kot sta PS Move in PS Eye.

Procesor v konzoli skrbi za vso obdelavo podatkov, ki jih pridobi od naprav, priključenih nanj.

Procesor je zasnovan na mikroprocesorju Cell. Mikroprocesor je skupni izdelek podjetij Sony, Sony Computer Entertainment, Toshiba in IBM. Tehnologija predstavlja most med procesorji, namenjenimi za računalnike, in procesorji, namenjenimi za grafične kartice.

Slika 2.16: Konzola PS3.

2.6.2.2 PlayStation EyeToy

Predhodnica PS Eye je bila naprava PS Eyetoy [27] (Slika 2.17).

Namenjena je kot dodatek h konzoli PS2.

Naprava je barvna digitalna kamera, podobna spletni kameri. Povezuje se preko USB 1.1 protokola. Sliko zajema v resoluciji 320 x 240 slikovnih točk.

Ima možnost, da z uporabo računalniškega vida pri procesiranju posnetih slik iz njih prepozna gibe teles. Vgrajen ima mikrofon, ki skrbi za zajem zvoka iz okolice.

Omogoča interakcijo z gibanjem, zaznavo barv in zvoka. Na napravi sta 2 LED lučki – modra in rdeča. Modra pomeni, da je naprava vključena.

Utripajoča rdeča lučka pomeni, da je premalo svetlo v prostoru, kjer je postavljena. Treba jo je bilo uporabljati v dobro osvetljenih prostorih. Ob izidu PS3 pa je izšla tudi prenovljena različica naprave, imenovana PS Eye.

Slika 2.17: EyeToy.

2.6.2.3 PlayStation Eye – PS Eye

Slika 2.18: PS Eye.

PS Eye [26] (Slika 2.18) je digitalna kamera, ki pri procesiranju slik uporablja računalniški vid, s tem pa prepoznavanje objektov, gibov, zvoka. Naprava je lahko samostojni senzor ali v povezavi z napravo PS Move.

Sestavljena je iz naslednjih delov:

- kamere,
- mikrofona – polja mikrofonov,
- stojala.

Povezuje in napaja se preko vrat USB 2.0.

Kamera

Kamera zajema sliko v dveh ločljivostih:

- VGA – 640 x 480 slikovnih točk pri 60 Hz,
- QVGA – 320 x 480 slikovnih točk pri 120 Hz.

Pri izdelavi kamere so upoštevali probleme, ki so nastale z njenim predhodnikom. To je bila predvsem svetlobna odvisnost. Za boljše zaznavanje okolice skrbi novi procesor, ki je narejen tako, da uporablja večje senzorje svetlobe. S tem pa se poveča občutljivost naprave pod TV svetlobo. Ima dva načina delovanja. Nastavljata se ročno preko rotacije objektiva kamere.
Ta dva načina sta:

- rdeča LED – 56° kot vidnega polja, ki se uporablja za bližnje posnetke,
- modra LED – 75° kot vidnega polja, ki se uporablja za oddaljene posnetke.

Podatke posreduje v dveh formatih:

- brez stiskanja podatkov – format podatkov RAW,
- s stiskanjem podatkov – format podatkov JPEG.

Mikrofon

Sestavljajo ga štirje mikrofoni v obliki kapsul. Konzola PS3 jih uporablja za različne namene. Eden od njih je zaznavanje govora v prostoru. Uporabljajo se tudi naslednji filtri:

- večkanalno izničevanje odmeva,
- večkanalno izničevanje šuma.

S tem nam omogoča uporabo aplikacij za prepoznavanje glasu – govora. Vsak kanal se procesira s 16 biti in z frekvenco vzorčenja 48 kHz.
Razmerje signala – šuma ima 90 dB.

2.6.2.4 Playstation Move – PS Move

Je upravljalnik [28], ki se uporablja v povezavi s PS Eye in omogoča upravljanje iger in naprave PS3.

(a) Move motion upravljalnik.

(b) Move navigator.

(c) Napajalna postaja.

Slika 2.19: Paket Move.

V paketu dobimo:

- PS Move motion upravljalnik (Slika 2.19a),
- PS Move navigator (Slika 2.19b),
- PS Move napajalno postajo (Slika 2.19c).

PS Move in PS Move navigator se povezujeta na PS3 preko protokola Bluetooth 2.0. Vgrajena ima tudi vrata USB mini B. Oblika spominja na mikrofon, le da ima na vrhu namesto membrane svetlikajočo kroglo. Krogla s pomočjo LED diod lahko sveti v vseh barvnih kombinacijah RGB.

Barvo krogle dinamično prilagaja s pomočjo barve okolice. Barvo dinamično določa na podlagi slik, ki jih zajema naprava Eye. Prilagodi jo tako, da ima svojo edinstveno barvo, ki je ni v okolici. S tem postane oznaka položaja oz. lege upravljalnika. Uporablja se za sledenje upravljalnika v vidnem polju naprave Eye. S tem ga tudi lažje umeščamo v 3D prostor.

Sestavljajo ga naslednji deli:

- senzorja premikanja:
  - 3-smerni senzor gibanja oz. pospeška,
  - 3-smerni senzor hitrosti vrtenja,
- lokacijski senzorji:
  - senzor magnetizma,
  - prepoznavanje objektov – v povezavi z Eye,
- 1 analogni gumb,
- 8 digitalnih gumbov,
- vmesnik Bluetooth.

Na konzolo lahko povežemo do 4 upravljalnike Move ali 2 Move in 2 Move navigatorja upravljalnika.

2.6.2.5 Knjižnica za NUI

PS Eye je na začetku uporabljal preproste metode računalniškega vida, kot so zaznava robov, barvno sledenje in maksimiranje obrazov. Ob najavi naprav Move in Kinect pa so izdali lastno knjižnico »Vision Library«. Velikost knjižnice je okoli 1–2 MB. Knjižnica vsebuje napredne metode računalniškega vida.

To so naslednje metode:

- napredna zaznava in analiza obraza:
  - predvsem natančna zaznava obraznih značilnosti (oči, ust, obrvi, nosa),
- sledenje glave s pomočjo računalniškega vida,
- prepoznavanje glasu:
  - metoda prepoznavanja glasu uporablja svojo knjižnico, ki naj bi podpirala 20 različnih jezikov.

Pri zaznavi obraza mislimo predvsem na aplikacijo, ki spremeni obraz uporabnika v neko animirano osebo. Pri tem sledi obrisu glave, odpiranju in zapiranju oči, zapiranju ustnic pri govorjenju. Uporabnika prepozna tudi po obliki obraza, starosti in spola, kar potem tudi uporabi pri animaciji.
Vision Libary je osnovna knjižnica, ki jo je razvilo podjetje Sony.

Podjetje Code Laboratories [8] je izdalo gonilnik in SDK za odprto uporabo PS Eye z omejitvami. Gonilnik je namenjen predvsem temu, da operacijski sistem pravilno zazna napravo. Z gonilniki se namesti tudi program, ki skrbi za upravljanje naprave. S pomočjo SDK pa lahko začnemo sami pisati svoje programe. Podpira naslednje programske jezike: C#, C++, Flash, Java, WPF, XNA.

Omejitve se nanašajo na število naprav, ki jih lahko uporabljamo hkrati. Ob nakupu plačljivih paketov pa omejitev števila naprav izgine. Paketi so razdeljeni glede na število naprav, ki jih želimo uporabljati hkrati. V pakete je vključena uporabniška podpora, napreden program, ki skrbi za upravljanje naprav, in dopolnjen SDK, ki vsebuje primere in hitra navodila za začetek uporabe.

2.6.3 Naprave Asus

2.6.3.1 Xtion Pro

Slika 2.20: Xtion Pro.

Xtion Pro [38] (Slika 2.20) je predhodnik Xtion Pro Live in prva naprava podjetja Asus, ki je bila namenjena upravljanju naprav z gibi. Ima vgrajen globinski senzor, sestavljen iz projektorja IR in senzorja IR. Napajanje in prenos podatkov poteka preko protokola USB 2.0. Zajem globinskih posnetkov poteka pri daljavah od 0,8 do 3,5 m. Nima vgrajenega mikrofona.
Vidno polje sestavljajo naslednji koti:

- horizontalni 58°,
- vertikalni 45°,
- po diagonali 70°.

Možne ločljivosti naprava so:

- VGA – 640 x 480 slikovnih točk pri 30 Hz,
- QVGA – 320 x 240 slikovnih točk pri 60 Hz.

2.6.3.2 Xtion Pro Live

Slika 2.21: Xtion Pro Live.

Xtion Pro Live [39] (Slika 2.21) ima vgrajeno RGB kamero, globinski senzor, ki je sestavljen iz IR projektorja in IR senzorja. Glede sestave senzorjev je zelo podoben Kinectu. Prenos podatkov in napajanje naprave potekata preko protokola USB 2.0. Zajem slike in globinskih posnetkov poteka pri daljavah od 0,8 do 3,5 m. Zajem zvoka poteka iz dveh mikrofonov, postavljenih ob sprednja robova naprave. Nima vgrajenega motorja za upravljanje nagiba naprave.
Vidno polje sestavljajo naslednji koti:

- horizontalni 58°,
- vertikalni 45°,
- po diagonali 70°.

Možne ločljivosti naprava so:

- VGA – 640 x 480 slikovnih točk pri 30 Hz,
- QVGA – 320 x 240 slikovnih točk pri 60 Hz,
- SXGA – 1280 x 1024 pri RGB sliki.

2.7 Primerjava naprav s senzorjem Kinect

Naprava / Primerjalni kriteriji	Kinect XBOX360 / PC	Playstation – Playstation Eye + Move	Konzola Wii	Xtion Pro Live
S čim upravljamo	S telesom	S telesom*, z Move motion upravljalnikom, z Move navigatorjem	Wii remote	S telesom
Koliko jih zazna/ koliko jih lahko aktivno sodeluje	XBOX360 8/4 PC 6/2	4/4	4/4	?/? **
Območje zajema v metrih od naprave	1,2–3,5 m, 0,8– 6 m, Kinect PC 0,4–6 m.	1–3 m.	Do 5 m za aktivno delovanje Wii remote, do 10 m jih zaznava.	0,8–3,8 m.
Vidno polje kamere	Horizontalni 57°, vertikalni 42°, po diagonali 70°.	Horizontalni 56° ali 75.°	Nima.	Horizontalni 58°, vertikalni 45°, po diagonali 70°.
Zvok / Mikrofoni	4 mikrofoni, povezani v polje	4 mikrofoni, povezani v polje	Wii remote ima vgrajen zvočnik	2 mikrofona, povezana v stereo način

Tabela 2.1: Primerjava 1.

* Ob uporabi pri določenih aplikacijah – drugače v kombinaciji z upravljalniki.

** Odvisno od aplikacije, ki upravlja napravo.

Naprava / Primerjalni kriteriji	Kinect XBOX360 / PC	Playstation – Playstation Eye + Move	Konzola Wii	Xtion Pro Live
Resolucija zajema slike / Hitrost zajema (Hz)	BARVNI ZAJEM RGB – 1280 x 960 (PC) slikovnih točk pri 12HZ (1280 x 1024 XBOX360) RYUV – 640 x 480 slikovnih točk pri 15 Hz VGA – 640 x 480 slikovnih točk pri 30 Hz GLOBINSKI ZAJEM 80 x 60, 320 x 240, 640 x 480 slikovnih točk pri 30 HZ	BARVNI ZAJEM VGA – 640 x 480 slikovnih točk pri 60 Hz QVGA – 320 x 480 slikovnih točk pri 120 Hz GLOBINSKI ZAJEM Nima globinskega zajema	Nima zajema	BARVNI ZAJEM SXGA – 1280 x 1024 pri RGB sliki. GLOBINSKI ZAJEM VGA – 640 x 480 slikovnih točk pri 30 Hz, QVGA – 320 x 240 slikovnih točk pri 60 Hz
Napajanje	Posebni napajalnik Y, ki se priključi na omrežje (12 V DC + 5 V USB)	Eye – Preko USB, Move – polni preko polnilne postaje, ima vgrajene akumulatorske baterije	Wii remote – preko navadnih AA ali akumulatorskih baterij Wii preko napajalnika	USB 2.0 5 V USB
Povezovanje	USB 2.0	Bluetooth USB 2.0 Wi-Fi	Bluetooth USB 2.0 Wi-Fi	USB 2.0

Tabela 2.2: Primerjava 2.

2.8 Možni SDK, knjižnice in gonilniki

Pri uporabi Kinecta se lahko poslužujemo raznih knjižnic in SDK, ki so nam na voljo. Te se razlikujejo predvsem po namembnosti uporabe in po tem, na katerih operacijskih sistemih delujejo. Predstavili bomo trenutno najbolj aktualne. Lahko jih delimo na dva načina:

- prvi način je glede gonilnikov, na katerih temeljijo SDK,
- drugi način pa glede na to, ali so odprtokodni ali licenčne narave.

Prvi način delitve:

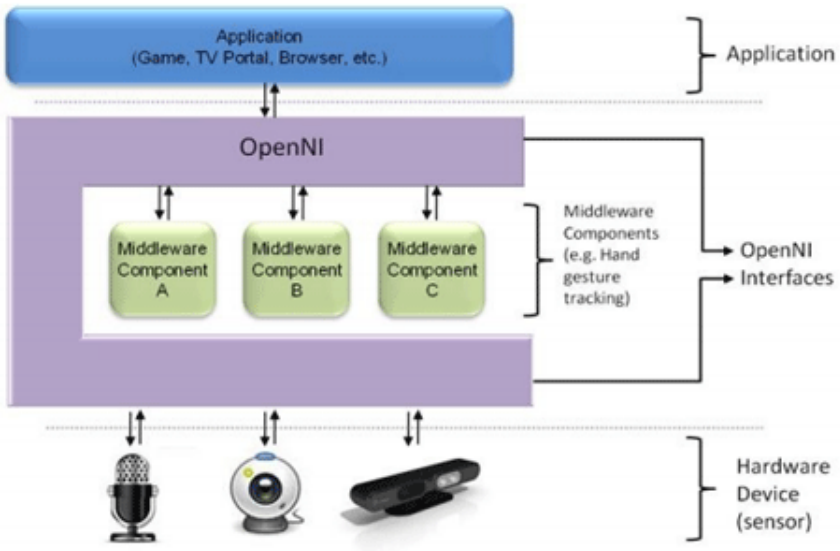
- Gonilnik PrimeSense
  - OpenNi – Nite
  - Microsoft Kinect SDK
- Odprtokodni gonilnik
  - OpenKinect/libfreenect
  - VRUI SDK

Drugi način delitve:

- Odprtokodne narave – GNU licenca
  - OpenNi – Nite
  - OpenKinect/libfreenect
  - CL NUI Platform
  - VRUI SDK
- Licenčne narave
  - Microsoft Kinect SDK

2.8.1 OpenNi

Istoimenska neprofitna industrijska organizacija [44] skrbi za promocijo uporabe NUI naprav. Skrbi tudi za vmesno programsko opremo (middleware), potrebno za delovanje NUI naprav. V njo so vključena naslednja podjetja: PrimeSense, Willow Garge, Side-Kick, Asus, AppSide. Njihov prvi cilj je bil razvoj odprtokodnega OpenNI ogrodja (angl. framework), ki nam prinaša programski vmesnik API. Z vmesnikom pa orodja za pisanje programov, ki uporabljajo NUI. Pri tem mislimo na kretnje, glasovne ukaze, gibe telesa in delov telesa. Namenjen je predvsem napravama Asus XtionPro in Asus XtionPro Live. Deluje pa tudi s Kinectom. OpenNI (različica 1.5.2.23, izdana 28. 12. 2011) in PrimeSense Sensor Modul za OpenNI/NITE (različica 5.1.0.41, izdana 28. 12. 2011) delujeta na Windows, Linux in Macintosh (podprti samo Intelovi procesorji in od OSX v10.6 naprej) operacijskih sistemih.



Slika 2.22: Arhitektura OpenNi.

OpenNi je ogrodje (Slika 2.22), ki povezuje napravo z vmesno programsko opremo. V njej se nahajajo različni gonilniki za naprave. Vmesna programska opreme je sestavljena iz različnih modulov, odvisno od potrebe uporabnika. S pomočjo modulov pa lahko naše aplikacije koristijo različne načine analiz. Ti moduli so:

- popolna analiza telesa – izdela paket informacij o telesu,
- analiza roke – generira informacije o lokacijah rok,
- zaznava kretenj – zaznavanje in uporaba kretenj,
- analiza okolja – koordinate okolja, identifikacija oblik.

Prednost OpenNi in njegovih modulov je v tem, da so odprtokodni, delujejo na več operacijskih sistemih in niso vezani na tip naprave.

2.8.2 OpenKinect/libfreenect

Slika 2.23: Logotip projekta OpenKinect.

Je odprta skupina ljudi, ki jih združuje to, da želijo Kinect čim bolj prilagoditi vsakdanji uporabi z računalnikom in drugimi tehnološkimi napravami. Vodja skupine je Josh Black. Skupina ima svojo spletno stran [23], na kateri so vsi podatki o projektu in tudi navodila za namestitev libfreenect. Svojo skupino predstavljajo tudi z logotipom (Slika 2.23). Programska oprema libfreekinect temelji na gonilniku libfreenect, ki nam ponuja programski vmesnik API. Preko API lahko dostopamo oz. upravljamo z/s:

- globinsko in barvno sliko,
- kontrolo motorja – nagib in izračun položajev x, y, z.
- kontrolo LED diode.

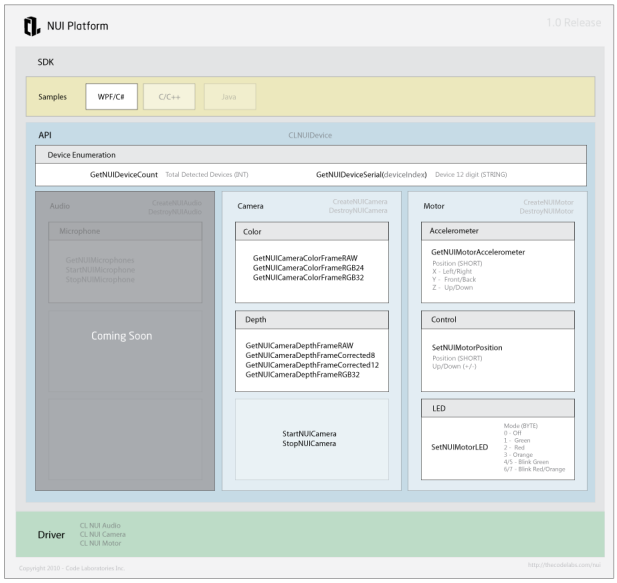
Ne podpira pa še zajema in upravljanja zvoka, pridobljenega iz mikrofonov. Gonilnik libfreenect je razvil Martin Hector. Je tudi prvi, ki je razvil in objavil odprtokodni gonilnik za Kinect.
Vsebuje tudi knjižnico OpenKinect analize, ki skrbi za komunikacijo z OpenKinect API-jem. Med komunikacijo analizira neobdelane podatke iz senzorja in jih pretvori oz. poveže v podatkovne strukture, ki nam predstavljajo informacije o določenih stvareh. Te stvari so podatki o skeletu, ozadju, motorju itd.
Deluje na operacijskih sistemih Linux, OS X in Windows.
Podpira programske jezike: C, C++, C#, Python, ActionScript, Java JNI, Java JNA, Javascript.

2.8.3 Platforma CL NUI

Podjetje Code Laboratories je izdalo SDK [7] (Slika 2.24), namenjen Kinectu. Vanj so vključeni odprtokodni gonilniki, programski vmesnik API, ki pa še ne izkorišča mikrofona in skeleta teles. Vsebuje naslednje možnosti:

- možnost uporaba več Kinectov hkrati,
- zajem globinske in barvne slike,
- kontrola motorja – nagib in izračun položajev x, y, z.
- kontrola LED diode – barve, utripanje.

Za programski jezik C # so v SDK priloženi tudi testni primeri uporabe programskega vmesnika. Trenutna različica je 1.0.0.120, ki je izšla 10. 12. 2010. Podpira naslednje 32-bitne in 64-bitne operacijske sisteme: Windows

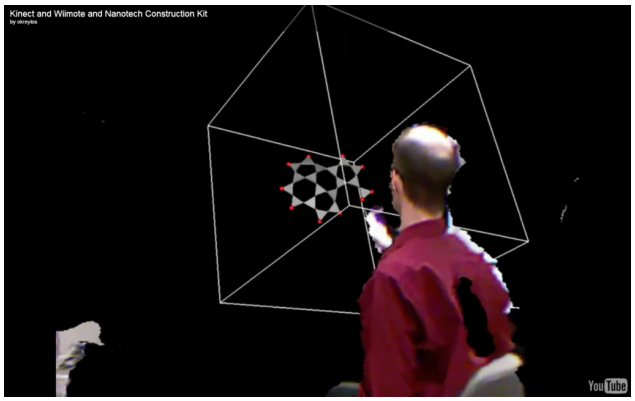


Slika 2.24: Arhitektura CL NUI platforme.

XP, Vista in Windows 7. Namenjen je za uporabo s programskima jezikoma C # in C++.

2.8.4 Orodje Vrui SDK – Vrui VR

Je odprtokodno orodje [24], ki ga je razvil Oliver Kreylos, zaposlen na univerzi Davis v Kaliforniji. Temelji na OpenGL programskem okolju. S pomočjo OpenGL generira 3D vsebino, ki se nato prikaže na 3D monitorjih, projektorjih itd. V SDK so vključene knjižnice in orodja, ki so namenjena hitremu razvijanju aplikacij za delo v navideznem oz. virtualnem 3D okolju. To so na primer uporabniški vmesniki v 3D prostoru, s katerimi upravljamo naprave itd. Projekt Kinect 3D Video Capture [16] temelji na orodju Vrui VR. Oliver Kreylos je kot del projekta razvil lastne gonilnike za Kinect in tudi svoj način spreminjanja navadnih slik v 3D slike. S pomočjo Kinecta, drugih naprav in svojega orodja posnete objekte postavi v 3D prostor, v njem pa potem posnete objekte po volji uporablja, upravlja, obdeluje in celo ustvarja nove objekte. Slika 2.25 prikazuje uporabo Kinecta v povezavi z upravljal-



Slika 2.25: Ustvarjanje oblik s pomočjo trikotnikov v 3D prostoru.

nikom Wii Remote in orodjem Nanotech Construction [58]. Oliver Kreylos se s pomočjo Kinecta postavi v 3D prostor, kjer z uporabo upravljalnika Wii Remote v orodju Nanotech Construction ustvari trikotnike in jih povezuje v poljubne oblike. Vse to je projicirano na 3D zaslon. Za ogled svojih kreacij uporablja posebna 3D očala. Orodje je pod GNU licenco in je na voljo vsakemu. Trenutna različica je Kinect 2.0. Za delovanje potrebuje nameščeno različico Vrui-2.2-003 ali novejšo. Deluje na operacijskem sistemu Linux.

2.8.5 Microsoft Kinect SDK

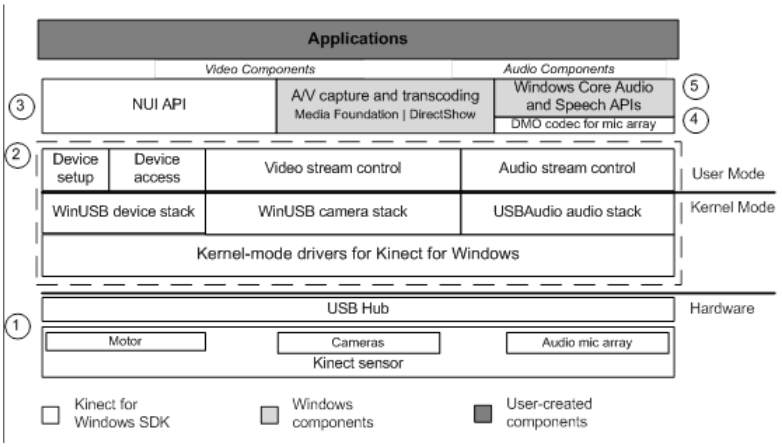
Microsoft je na odprtokodne gonilnike in na NITE odgovoril s tem, da je izdal svoj SDK. Zgodovina izdaj:

- SDK različica beta – junij 2011,
- SDK različica beta refresh – avgust 2011,
- SDK različica beta2 – november 2011,
- SDK različica 1 – februar 2012,
- SDK različica 1 (različica 1.0.3.191) - izšla 2. 5. 2012,
- SDK različica 1.5 (različica 1.5.2.331) in Toolkit 1.5.1 (različica 1.5.0.145) – izšla dne 18.5.2012.

Z izidom SDK različice 1 februarja 2012 in hkratno izdajo Kinecta za PC je Kinect SDK prešel v komercialno uporabo. Uporabo SDK različice Beta 2 s Kinect XBOX 360 so podaljšali do 16. junija 2016. Pogoj za uporabo je, da se razvija in uporablja aplikacije samo za nekomercialne namene. Je namenjena predvsem za testiranje, raziskovanje in začetno uporabo z XBOX Kinectom.

SDK različica 1 je namenjena za komercialno uporabo in ni odprtokodna. Microsoft je lastnik SDK. Ob uporabi se strinjamo z določenimi pogoji uporabe. Prilagojen je za napravo Kinect PC. Uradno na novejših SDK-jih ne podpirajo razvoja aplikacij za XBOX Kinect, kljub temu pa se da razvijati aplikacije zanj, s precej omejitvami. Pri omejitvah mislimo predvsem na to, da ne bodo delovali določeni testni primeri, ki so namenjeni PC verziji Kinecta. Microsoft svetuje vsem, da kupijo Kinect PC verzijo, zaradi splošnega prehoda večine uporabnikov na novejšo verzijo. S tem pa se usmeri razvoj aplikacij le na PC verzijo Kinecta, ki je bolj napredna.

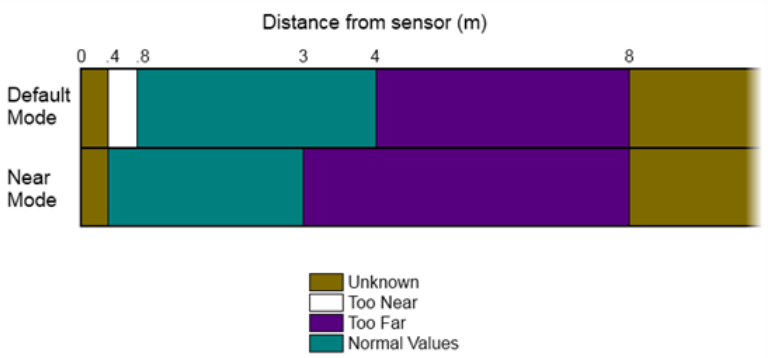
Arhitektura SDK (Slika 2.26) nam razkriva, da ima realizirane skoraj vse komponente, ki jih rabimo za upravljanje s Kinectom. SDK skrbi za gonilnike, logiko povezovanja naprave z računalnikom, upravljanje naprave, upravljanje različnih podatkovnih tokov. Preko programskega vmesnika API pa omogoča našim aplikacijam upravljanje naprave. Realizirani so tudi kodeki – filtri za mikrofone. V SDK niso realizirani zajem, kodiranje videa in zvoka. Ne vsebuje tudi programskega vmesnika API za govor in zvočnega jedra. Za te poskrbi operacijski sistem.



Slika 2.26: Arhitektura Microsoft Kinect SDK.

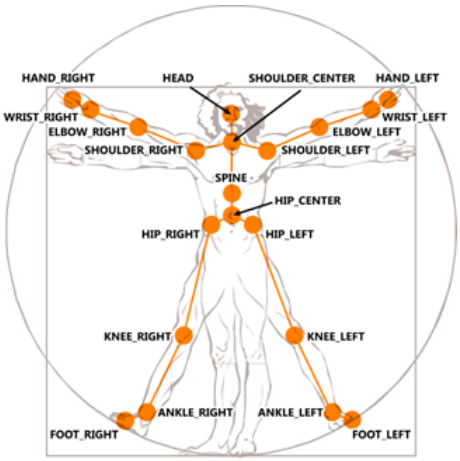
Uradno podpira samo operacijske sisteme Windows 7 in Windows Embedded Standard 7. Podpora zanj bo vgrajena v novi operacijski sistem Windows 8. SDK različica 1 nam prinaša podporo za hkratni priklop in uporabo štirih Kinect naprav. Izboljšali so spremljanje skeleta, izboljšali natančnost prepoznavanja glasa. Posodobili so programski vmesnik API. Odpravili so veliko težav s stabilnostjo, ki so se pokazale ob uporabi SDK beta2.

Kinect PC pa nam prinaša uporabo posebnega načina globinskega senzorja, ki prepoznava objekte že 40 cm pred napravo (Slika 2.27). Imenuje se bližnji način (angl. Near Mode).V tem načinu je maksimalno območje uporabe 3 m ob tem, da je natančnost v območju 2 m 100 %. To so omogočili s spremenjeno logiko v napravi. Uporabljamo lahko oba načina.



Slika 2.27: Dva načina globinskega senzorja.

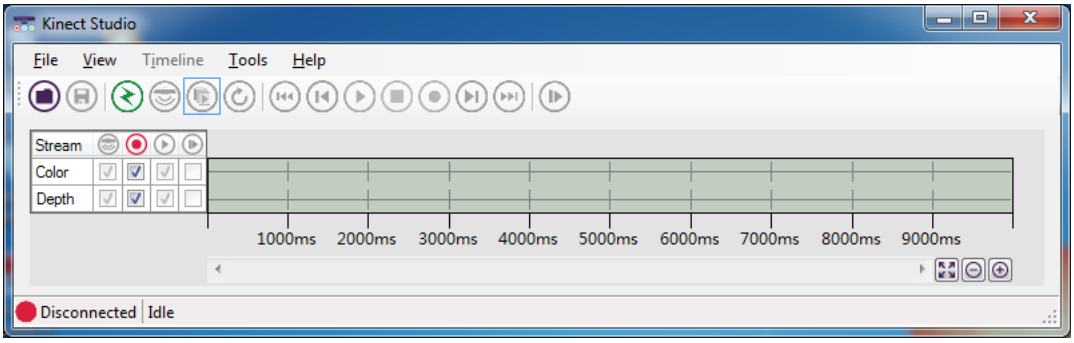
Kinectova logika in SDK nam omogočata relativno postavitev skeletnih točk na zaznano telo. Na sliki 2.28 so prikazane vse skeletne točke (vseh je 20), ki se uporabljajo pri skeletu.



Slika 2.28: Skeletne točke.

2.8.5.1 Kinect SDK različica 1.5

SDK je bil razdeljen na dva namestitvena paketa - SDK jedro in razvojno orodje (angl. toolkit). V prvem delu so vključeni osnovni gonilniki za Kinect, izvajalnik kode - izvajalno okolje in osnovni SDK. Primeri uporabe, razna uporabna orodja in vsa dokumentacija se nahajajo v drugem delu. Prinaša nam podporo glasovnega prepoznavanja 11 jezikov, možnost sledenja skelete v sedečem položaju (sledi samo zgornjim 10 točkam). Vsebuje SDK namenjen sledenju obraza (angl. Face Tracking SDK), ki je namenjen zaznavanje obraznih značilk kot so oči, nos, usta in usmeritev obraza (pri PC verziji Kinecta). Vsebuje tudi aplikacijo Kinect Studio s katero lahko zajemamo tokove (Slika 2.29). Odpravili so tudi določene težave s stabilnostjo. Omogočili so boljšo sinhronizacijo med barvnim in globinskim tokom. Izboljšali so kvaliteto zajema RGB slike.



Slika 2.29: Aplikacija Kinect Studio.

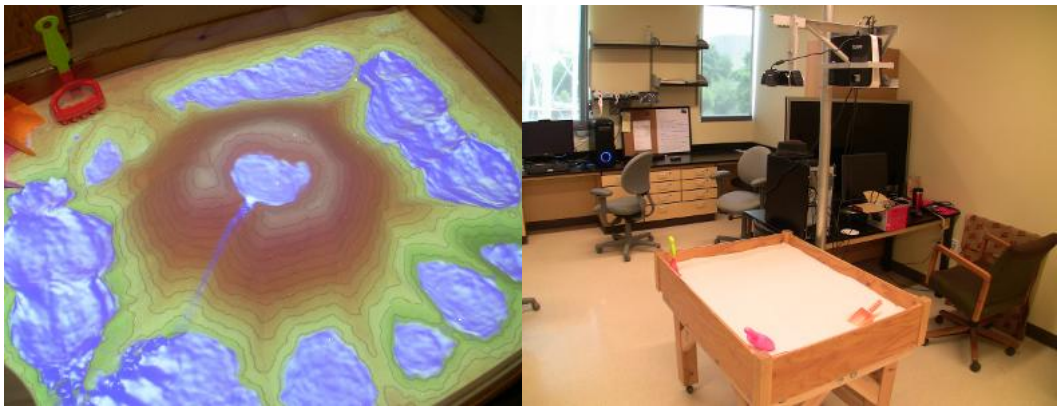
Novi SDK še bolje izkoristi oziroma je še bolj prilagojen PC verziji Kinecta.

2.9 Zanimivi projekti realizirani s pomočjo Kinecta

Tekom dveh let obstoja Kinecta je bilo razvito veliko aplikacij namenjenih za Kinect. Veliko projektov je predstavljeno na spletnih straneh KinectHacks [19], Code4Fun [9] in kinect.dashhaks [18]. V nadaljevanju bomo predstavili projekte, ki so nam bili najbolj zanimivi.

2.9.1 Augmented Reality (AR) Sandbox

Enega od projektov, ki ga je ustvaril Oliver Kreylos smo bolj natančno opisali že pod točko 2.8.4. S strani Oliverja in ostalih članov projektne skupine sledi zanimiv projekt navidezno resnični peskovnik (angl. Augmented Reality (AR) Sandbox) (Slika 2.30a). Naprava je sestavljena iz peskovnika, Kinecta in digitalnega projektorja. S spreminjanjem oblike peska v peskovniku ustvarjajo virtualno topografski zemljevid okolja. V tem okolju pa so lahko hribi, gorovja, doline, reke, jezera in vulkani z lavo. To poteka s pomočjo zajema 3D slike preko Kinecta in simulacij izvedenih s pomočjo orodja Vrui SDK. Za prikaz slike na peskovniku skrbi digitalni projektor. Primer uporabe je prikaza na sliki 2.30a.



(a) Primer uporaba peskovnika. (b) Peskovnik.

Slika 2.30: Primera peskovnika.

Idejo za projekt so dobili s strani čeških raziskovalcev, ki so sami s pomočjo Kinecta razvili preprosti navidezni resnični peskovnik [40].

Obstaja tudi podobni projekt Nizozemcev imenovan Mimicry [47]. Pri tem projektu zajemajo oblike iz peskovnika v 3D prostor in jih raziskujejo kot neka 3D raziskovalna igra. S projekcijo na peskovnik pa prikazujejo ustvarjeni 3D prostor in naše premike znotraj prostora.

Na uradni straniprojekta Augmented Reality (AR) Sandbox so prikazane slike postavitve prototipa [3]. Ob njih je opis celotne postavitve naprava in njenega delovanje. Posneli so tudi dva filma [4], ki prikazujeta uporabo peskovnika. Bolj podrobno je projekt opisan na uradni spletni strani [2].

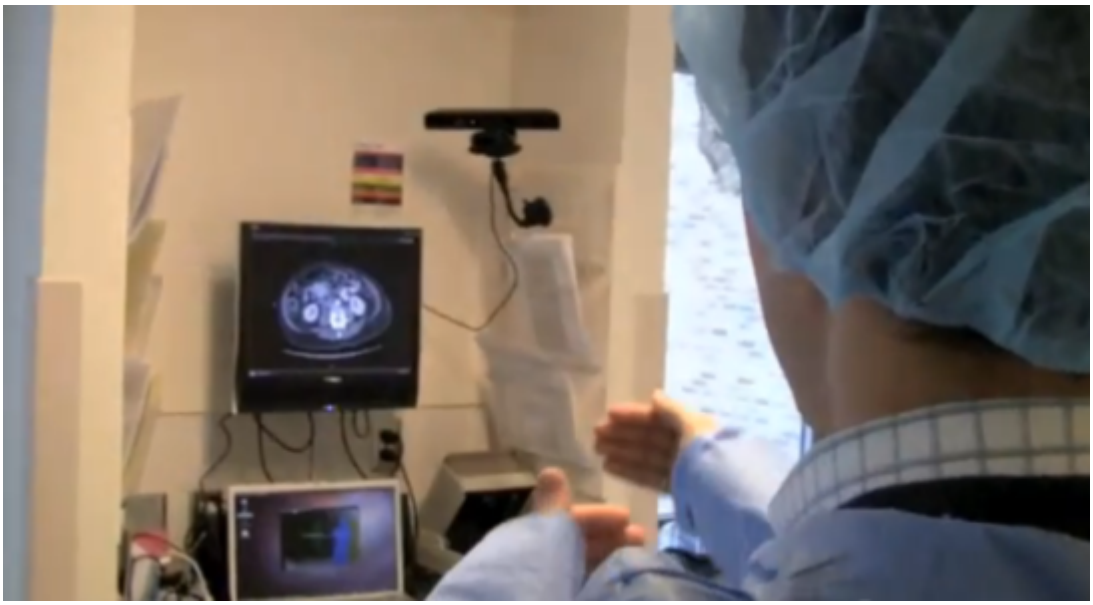
2.9.2 Kinect + 3D TV = Virtual Reality

Slika 2.31: Primer postavitve sistema Virtual Reality.

Sistem (Slika 2.31) je kombinacija Kinecta, Panasonic 3D plazma televizorja in aplikacije oziroma programske opreme razvite s strani dr. Roberta Kooima. Dr. Roberta Kooima je docent na oddelku za računalništvo univerze v Louisiani [42]. Namen postavljenega sistema je, da s pomočjo Kinecta spremlja položaj uporabnika. Ob spremljanju prilagaja 3D projekcijo in virtualno okolje uporabnikovem položaju glave - očem. Pomen tega je, da lahko spreminja zorni kot prikaza v virtualnem okolju. Na primer, če se uporabnik skloni vidi objekt, ki je na sliki iz spodnjega zornega kota. Uporabnik ima možnost, s pomočjo roke, upravljati elemente v virtualne okolju. Za razvoj aplikacije so uporabili OpenNi ogrodje in lastno razvito skriptno okolje Electro VR [43]. Aplikacija se izvaja na operacijskem sistemu Ubuntu. Prikaz 3D slike je izrisan s pomočjo OpenGL in grafične kartice Nvidia GTX 470. Posnel je tudi posnetek [41], ki prikazuje uporabo sistema.

2.9.3 Projekti v zdravstvu

2.9.3.1 Xbox Kinect in the hospital operating room

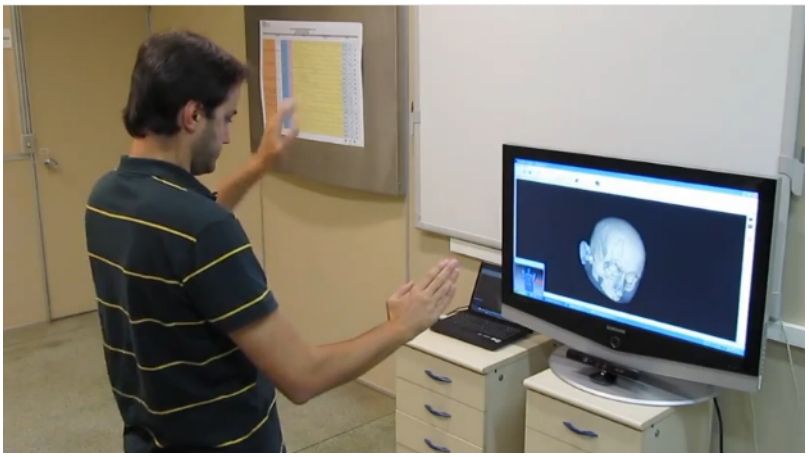


Slika 2.32: Primer upravljanja posnetkov s Kinectom.

Bolnišnica SunnyBrook v Torontu je s pomočjo študentov Univerze v Torontu razvila lastno aplikacijo, ki upravlja prikaz slik posnetih z magnetno resonanco (angl. CT, MRI or CAT pictures) s pomočjo Kinecta. Uradna izjava za javnost se nahaja na njihovi spletni strani [56]. Nekaj izjav o uporabi Kinecta s strani kirurgov je zbranih na spletnem dnevniku Monice Matys [57]. Posneli so tudi film [55], ki prikazuje upravljanje posnetkov s kretnjami. Primer uporabe je prikazan na sliki 2.32.

2.9.3.2 Gesture Interface using Kinect for Medical Imaging Visualization in Surgeries

Podoben projekt kot v točki 2.9.3.1 so razvili v ustanovi Renato Archer Center for Information Technogy – CTI v mestu Campinas v Braziliji [35]. S pomočjo Kinecta upravljalo lastno razviti program InVesaluis (Slika 2.33). Program je odprtokodni namenjen generiranju, gledanju in upravljanu medicinskih 3D slik iz formata 2D DICOM. Format izhaja iz posnetkov zajetih z magnetno resonanco (angl. MRI or CT scans).



Slika 2.33: Primer upravljanja aplikacije InVesaluis.

Posneli so tudi film [54], ki prikazuje uporabo programa s Kinectom. Več o projektu in programu je napisano na uradni strani projekta [46].

2.9.4 Projekti za slabovidne oziroma slepe ljudi

2.9.4.1 NAVI – Navigational Aids for the Visually Impaired

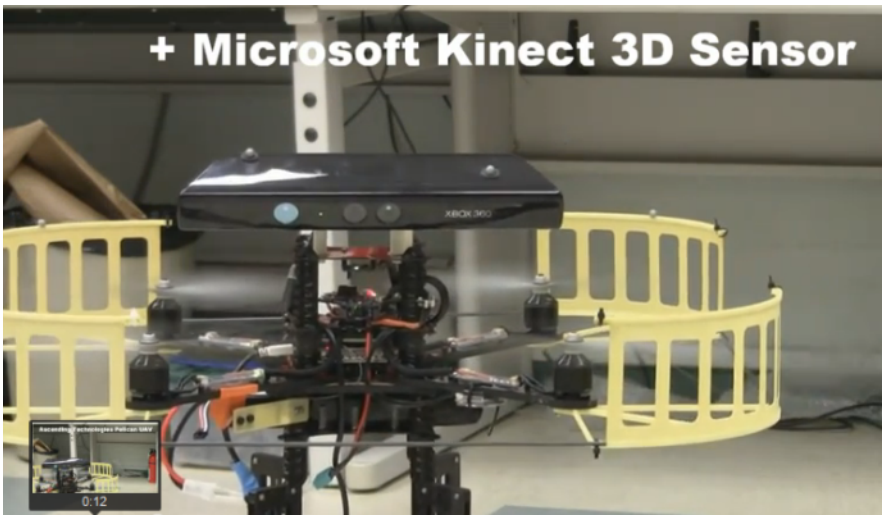
Podiplomska študenta Michael Zöllner in Stephan Huber iz univerze Konstanz sta razvila testni sistem (Slika 2.34) za slabovidne oziroma slepe ljudi, ki bi jim pomagal pri navigaciji v notranjih prostorih stavb. Palica služi kot osnovni pripomoček za slabovidne in slepa. Ima kratek radij in deluje samo za stvari, ki so na tleh. Za povečanje radija sta s pomočjo napravo Kinect razvila sistem, ki skrbi za navigacijo po hodnikih. Na čelado sta namestila Kinect in ga napajala z desetimi 1,2 V baterijami. Izdelala sta tudi lasten vibracijski pas s katerim opozarjata v katero smer mora zaviti oseba. Za pas sta uporabila tri »LilyPad« motorčke, jih za večji učinek vstavila v pokrovčke plastenk ter prilepila na pas. Motorčke sta povezala na vezje Arduino 2009. Vezje v pasu in Kinect sta preko USB vhodov povezana na prenosnik, ki je bil nameščen v lastno prilagojen nahrbtnik.

Slika 2.34: Test sistema NAVI.

S pomočjo aktivnih označb orodja AR sta označila pot po stavbi. Njuna aplikacija s pomočjo RGB kamere označbe spremlja v realnem času. Osebi ponuja glasovna navodila kam mora zaviti. Navodila se spreminjajo glede na oddaljenost osebe od označbe. Oddaljenost ugotavlja s pomočjo globinskega senzorja. Aplikacija je napisana v programskem jeziku C#.NET. Za ogrodje sta uporabila MangedOpenNI, za sledenje iz izdelavo aktivnih označb sta uporabila ARToolkitPlus. Za sintezo glasu sta uporabila Microsoft Speech API. Vse vhodne tokove sta združila s pomočjo Reactive Extensions za .NET Več o projektu na uradni strani univerze [48]. Posnela sta lastni film [49], ki prikazuje test sistema. Koda projekta in seznam uporabljenih programskih paketov pa sta objavila na codeplex strani [50].

2.9.5 Projekti v robotiki

2.9.5.1 Quadrotor + Kinect



Slika 2.35: Na Quadrotor nameščeni Kinect.

Je projekt razvit s strani podiplomskega študenta Patricka Bouffard z univerze Barkeley v Kaliforniji [51]. Kinect je namestil na helikopter imenovan »quadrotor« (Slika 2.35), ki je naprava AscTec Pelican [45]. Kinect se napaja iz baterije nameščene v helikopter. Aplikacija bazira na ogrodju libfreenect in ROS (angl. Robot Operating System) gonilnika za Kinectovo RGB-D kamero [52]. Gonilnik je odprtokodni in je izpeljan iz gonilnika, ki ga je ustvaril Hector Martin. Z logiko v aplikaciji helikopter samodejno leta po prostorih in si nastavlja optimalno višino lebdenja. Izogiba se tudi oviram na poti. Lahko tudi leti po poti, ki smo mu jo sprogramirali. Če pri ročnem načinu leta naleti na oviro se ustavi in lebdi na mestu. Ko ovira izgine nadaluje pot. Za tekmovanje ROS 3D je bil izdelan paket starmac-ros-pkg [53], ki vsebuje osnovna programska orodja za delo s Quadrotorjem in Kinectom.

2.9.5.2 Project Icarus

Slika 2.36: Prikaz naprave uporabljene v projektu Icarus.

Projekt [29] je ideja nizozemskih študentov odelka Gamedesign and Mechatronica izobraževalnega centra ROC Friese Poort Drachten. Osnovni namen projekta je simulator letenja. S pomočjo Kinecta in položaja rok, preko lastno razvite aplikacije, krmilijo servo motorje v napravi. Ti skrbijo za nagib naprave in s tem položaja osebe. Na napravo je nameščen 55"LCD monitor, ki skrbi za prikaz virtualnega sveta. Virtualni svet se odziva na ukaze prejete s strani Kinecta. Naprava je prikazana na sliki 2.36. Posneli so tudi filma, ki prakzujeta test sistema. Prvi film [30] še nima nameščenega LCD zaslona. Pri drugem filmu [31] pa je že nameščen LCD zaslon, ki prikazuje virtualno okolje.

Poglavje 3

Razvoj aplikacije

3.1 Postavitev delovnega okolja

Kot smo omenili, Microsoft SDK različice 1 deluje samo na operacijskih sistemih Windows 7 in Windows Embedded Standard 7. Kompatibilen bo tudi z novim operacijskim sistemom Windows 8.
Zmogljivosti prenosnika, na katerem bo potekal razvoj aplikacije:

- Intel® Core™ i5-2430 M (2.40 GHz, 3 MB L3 predpomnilnika), do 3.00 GHz,
- 4 GB DDR3 1333MHZ,
- 640 GB SMART SATAII 5400 rmp,
- Windows 7 Professional x64, Microsoft Visual Studio 2010 Express C# in Microsoft Office 2010 Professional,
- Microsoft Kinect SDK in knjižnica funforcoding.

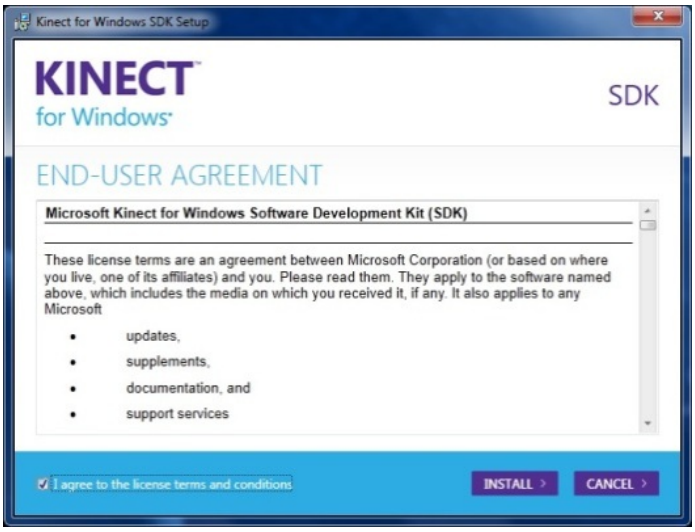
Z uradne strani [11] je treba naložiti namestitveni paket KinectSDK-v1.0-Setup.exe. Paket vsebuje gonilnike za 32 in 64-bitni operacijski sistem Windows 7. Vsebuje tudi paket Microsoft Speech Platform – Runtime (različica 11), ki je potreben za uporabo priloženih primerov uporabe mikrofonov.

Če želimo razvijati aplikacije, ki uporabljajo prepoznavanje besed, glasovnih ukazov, moramo namestiti paket Microsoft Speech Platform – Software Development Kit (SDK) (različica 11).

Za pravilno prikazovanje primerov v programskem jeziku C++ pa potrebujemo še Microsoft DirectX® SDK in DirectX End-User Runtime. Prvi nam prinese veliko orodij in primerov za izdelavo multimedijskih aplikacij s pomočjo Direct3D, drugi pa zadnje posodobitve za komponente D3DX, prevajalnik HLSL, XInput, XAudio in Managed DirectX 1.1.

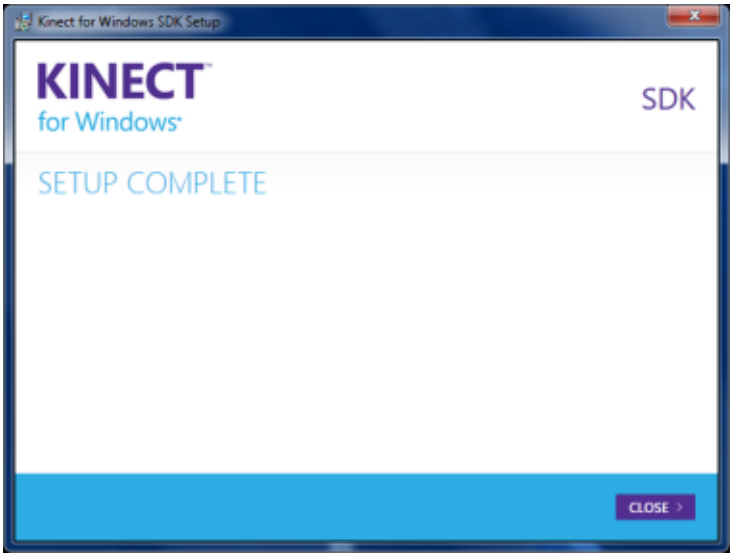
Namestitveni postopek je precej enostaven, ker so v SDK različice 1 združili vse potrebne programske pakete za delo s Kinectom.

Pred namestitvijo SDK različice 1 je potrebno odstraniti stari SDK in platforme za govor. Kinect mora biti med namestitvijo izklopljen iz računalnika. Ko zaženemo namestitveni paket KinectSDK-v1.0-Setup.exe se nam prikaže okno (Slika 3.1). V oknu se nam prikaže besedilo licenčne pogodbe, s katero



Slika 3.1: Namestitev Kinect SDK za Windows.

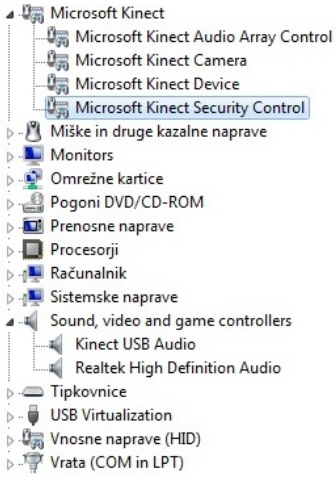
se zavežemo k upoštevanju določenih pogojev. Namestitveni postopek traja nekaj časa. Namestijo se SDK, Kinectovi gonilniki, Microsoft platforma za govor.



Slika 3.2: Konec namestitve.

Ob koncu namestitve se pokaže okno (Slika 3.2). Zdaj lahko Kinect priklopimo na računalnik. Ob priklopu se začnejo nameščati gonilniki, ki so bili nameščeni z SDK. Ob pravilni namestitvi se v upravljalniku strojne opreme pokažejo naslednje naprave (Slika 3.3) :

- Microsoft Kinect Audio Array Control,
- Microsoft Kinect Camera,
- Microsoft Kinect Device,
- Microsoft Kinect Security Control,
- Microsoft Kinect USB Audio.



Slika 3.3: Upravljalnik naprav.

3.1.1 Nadgradnja na SDK različico 1.5

Z uradne strani [11] je potrebno naložiti namestitveni paket KinectSDK-v1.5-Setup.exe [12]in KinectDeveloperToolkit-v1.5.1-Setup.exe [13]. Kot je bilo omenjeno v točki 2.8.5.1 so razdelili SDK na dva dela. Prvi namestitveni paket samodejno poskrbi za pravilno nadgradnjo iz različice 1 na 1.5. Ob primeru, da nimamo nameščene različice 1 pa deluje kot samostojna namestitev. Drugi namestitveni paket namesti praktične primere, dodatna orodja in vso dokumentacijo za delo s Kinectom. Za hitro in enostavno delo s Kinectom in .NET aplikacijami je potrebno namestiti Microsoft.Kinect.Toolkit, ki se nahaja pod brskalnikom »Developer Toolkit Browser v1.5.1 (Kinect for Windows). Brskalnik se namesti z namestitvijo drugega paketa. Prav tako se v njem nahaja API za sledenje obraza (»Microsoft.Kinect.Toolkit.FaceTracking«). Ko želimo uporabljati različne jezike pri prepoznavanju govora jih moramo namestiti z njihove spletne stran [14]. Bolj podroben opis sprememb je na uradni MSDN strani [15].

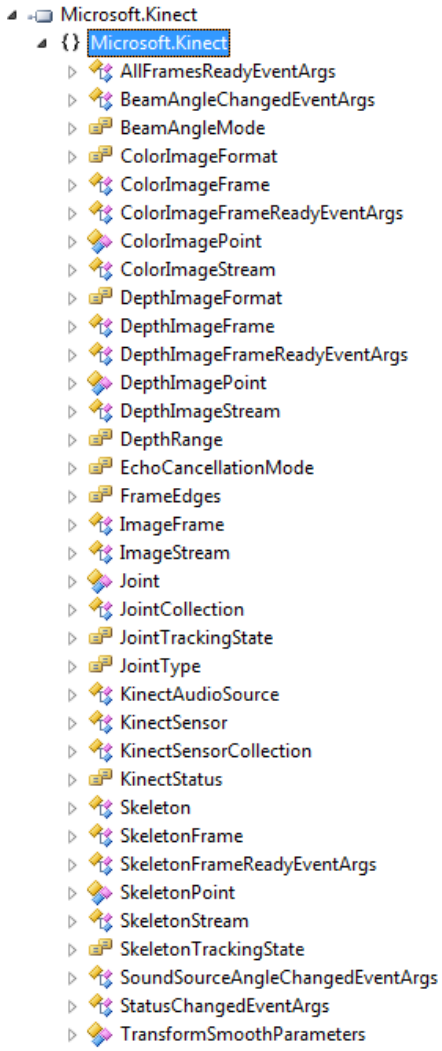
3.2 Aplikacija Upravljalnik Kinect

3.2.1 Ideja oziroma namen aplikacije

Kot smo omenili že v uvodu, je osnovni namen preproste aplikacije spoznavanje in uporaba različnih tokov naprave Kinecta. Pri tem ne bomo uporabljali mikrofonov oz. zvočnega toka podatkov. Osredotočili se bomo predvsem na raven upravljanja aplikacij z določeni gibi, ki se pretvorijo v ukaze. Ukazi se nato posredujejo trenutno aktivni aplikaciji. Aplikacija naj bi bila modularna z možnostjo hitrega dodajanja možnosti in upravljanja različnih aplikacij. Za ta namen bo morala biti tudi prilagodljiva oziroma univerzalna.

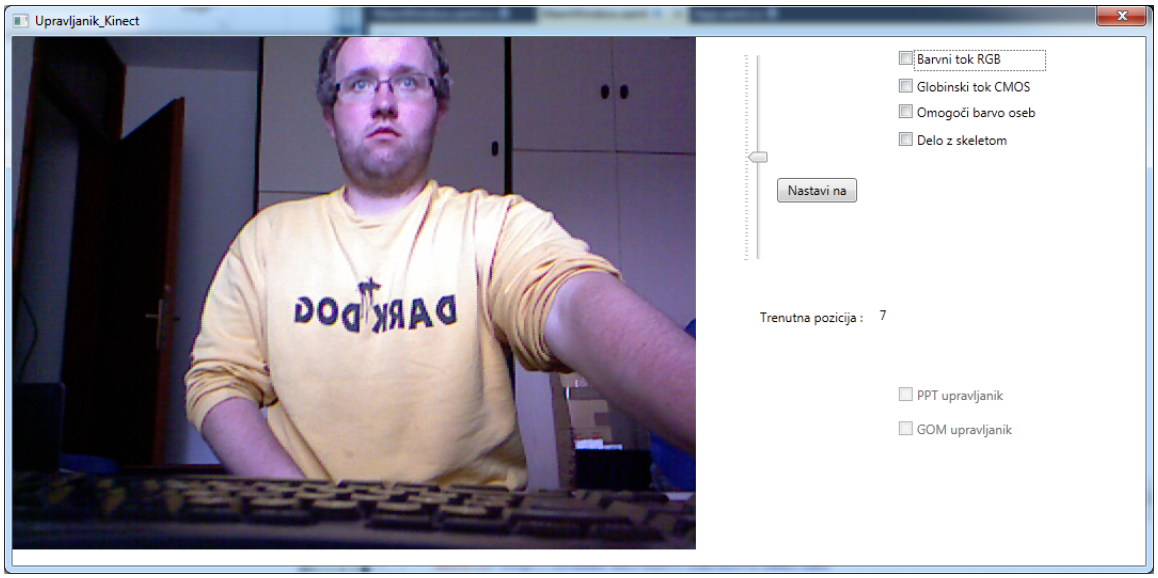
3.2.2 Razvoj aplikacije Upravljalnik Kinect

Za začetek razvoja potrebujemo Microsoft Visual C# 2010 Express. Je brezplačno orodje, ki omogoča razvoj aplikacij v programskem jeziku C#. Za vključitev in delo Kinecta poskrbi Kinect SDK. V našem projektu moramo navesti referenco na knjižnico, ki skrbi za Kinect. V kodi programa navedemo, da uporabljamo »Microsoft.Kinect« (Slika 3.4).

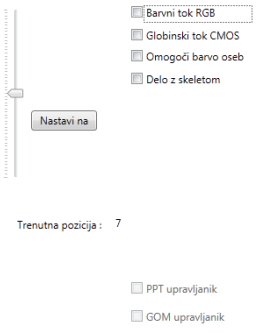


Slika 3.4: Referenca na knjižnico »Microsoft.Kinect«.

Za hitrejši začetek so Kinect SDK priloženi razni vodiči, informacije o SDK, SDK pomoč in primeri uporabe. Aplikacija Upravljalnik Kinect (Slika 3.5) je sestavljena iz dveh večjih delov. Prvi del je prikazovalnik slike. Prikazuje tokove glede na izbiro potrditvenih polj. Drugi del (Slika 3.6) je ukazni del, s katerim spreminjamo načine prikaza slike, naklon Kinecta, uporabo skeletnih točk in upravljanje aplikacij s Kinectom.



Slika 3.5: Aplikacija Upravljalnik Kinect.



Slika 3.6: Izbirne možnosti aplikacije.

V nadaljevanje bomo predstavili kodo programa, ki se skriva za drugim delom aplikacije. Iz Kode 3.1 so razvidne definicije spremenljivk, ki jih bomo uporabljali tekom aplikacije. Kinect senzorju smo dali ime naprava, določili smo maksimalno in minimalno polje zajema globinske slike. Spremenljivke tipa »bool« smo uvedli za preprosto zaščito, ki bo skrbela za enkratno pošiljanje ukaza. Konstante tipa »float« nam bodo v pomoč pri delu z določevanjem kretenj za ukaze. Na začetku tudi fiksno določimo spremenljivki za formata barvne in globinske slike. Odločili smo se, da jih definiramo že na začetku, da točno vemo, katere formate uporabljamo. Polje skeletov oziroma teles fiksno določimo na velikost 6, ker novi Kinect lahko hkrati spremlja 6 teles. V resnici bomo tekom programa spremljali in upravljali samo z enim skeletom.

```
KinectSensor naprava; //definiramo ime Kinecta

const float maksglobina = 4000; // MAX ODDALJENOST
const float minglobina = 800; // MIN ODDALJENOST
//RAZLIKA V RAZDALJAH UPORABLJENO ZA ODTENKE
const float razlikaglobina = maksglobina - minglobina;

//ZASCITA DA NE PRIDE DO POSILJNJA VECKRATIH UKAZOV
bool aktivnaleva = false;
bool aktivnadesna = false;
bool gor = false;
bool dol = false;
bool spredaj = false;
bool zadej = false;

//FAKTOR ODALJENOSTI OD NEKE TOCKE
const float faktorrazdaljeX = 0.4f;
const float faktorrazdaljeY = 0.4f;
const float faktorrazdaljeZ = 0.4f;

//Format barvne slike
ColorImageFormat barvni_format = ColorImageFormat.RgbResolution640x480Fps30;
//Format CMOS slike
DepthImageFormat globinski_format = DepthImageFormat.Resolution320x240Fps30;

// SLEDI MAKSIMALNO 6 OSEBAM oziroma SKELETOM - SDKv1 --- SDKbeta2 do 8 skeletom
// Tabela skeletov
Skeleton[] vsa_telesa = new Skeleton[6];
```

Programska koda 3.1: Definicije spremenljivk.

Koda 3.2 se izvede, ko se naloži glavno okno aplikacije. Ob tem preveri, ali je število Kinectov večje od nič, če to ne drži, nam vrne napako o nepovezanosti Kinecta. Ko imamo priklopljen Kinect na računalnik in je v napajanju ter ima vzpostavljeno povezavo (utripa zelena lučka), se omogočijo tokovi, s

katerimi bomo upravljali. Upravljali bomo s skeletom, barvno sliko in globinsko sliko. Ob tem se vzpostavijo upravljalniki dogodkov za vse tri tokove. Vzpostavili bomo tudi generičen upravljalnik dogodkov, ki lahko upravlja z vsemi tokovi hkrati. Po vzpostavitvi pokličemo še funkcijo »skrijskelet()«, ki nam skrije točke skeleta in funkcijo »naprava.Start()«, ki zažene Kinect. Zagon naprave je lepo viden ob zagonu IR projektorja, ker se mu leča obarva rdeče. V spremenljivki »label1« se izpiše naklon naprave.

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    if (KinectSensor.KinectSensors.Count > 0)
    {
    naprava = KinectSensor.KinectSensors[0];

    if (naprava.Status == KinectStatus.Connected && !(naprava.Status == KinectStatus.NotPowered) &&
          !(naprava.Status == KinectStatus.Initializing))
    {
        naprava.SkeletonStream.Enable();
        naprava.ColorStream.Enable(barvni_format);
        naprava.DepthStream.Enable(globinski_format);
        naprava.AllFramesReady += new EventHandler<AllFramesReadyEventArgs>(naprava_AllFramesReady);
        naprava.ColorFrameReady += new
              EventHandler<ColorImageFrameReadyEventArgs>(naprava_ColorFrameReady);
        naprava.DepthFrameReady += new
              EventHandler<DepthImageFrameReadyEventArgs>(naprava_DepthFrameReady);
        //naprava.SkeletonFrameReady +=new
              EventHandler<SkeletonFrameReadyEventArgs>(naprava_SkeletonFrameReady);
        skrijskelet();
        naprava.Start();
        label1.Content = naprava.ElevationAngle.ToString();
        //label1.Content = naprava.UniqueKinectId.ToString();
        }
        else if (naprava.Status == KinectStatus.NotPowered)
        {
        System.Windows.Forms.MessageBox.Show("Naprava nima napajanja-Vkljuci napajanje", "NAPAKA");
        Close();
        }
        else if (naprava.Status == KinectStatus.Initializing)
        {
        System.Windows.Forms.MessageBox.Show("Naprava se pripravlja", "VZPOSTAVLJANE");
        }
    }
    else
    {
    System.Windows.Forms.MessageBox.Show("Naprava ni povezana na racunalnik", "NAPAKA");
    Close();
    }
}
```

Programska koda 3.2: Inicializacija tokov in zagon Kinecta.

S pomočjo preverjanj poskrbimo tudi za obvestilo o napajanju naprave in inicializaciji naprave.

```
void naprava_ColorFrameReady(object sender, ColorImageFrameReadyEventArgs e)
{
    if (checkBox1.IsChecked == true)
    {
        using (ColorImageFrame barvna_slika = e.OpenColorImageFrame())
        {
            if (barvna_slika == null)
            {
                return;
            }
            byte[] pixels = new byte[barvna_slika.PixelDataLength];
            barvna_slika.CopyPixelDataTo(pixels);
            int sirinaslike = barvna_slika.Width * 4;
            image1.Source = BitmapSource.Create(barvna_slika.Width, barvna_slika.Height, 96, 96,
                PixelFormats.Bgr32, null, pixels, sirinaslike);
        }
    }
}
```

Programska koda 3.3: Funkcija za prikaz barvne slike.

Funkcija »naprava ColorFrameReady« (Koda 3.3) skrbi za pravilen prikaz RGB slike na modulu »image1«. Naprej napovemo, da bomo uporabljali barvne slike – okvirje. Zatem sledi preverba ob izgubi slik – okvirjev. Ustvarimo polje oziroma tabelo »pixels«, ki hrani vrednosti slik. S pomočjo metode »CopyPixelDatoTo« jo napolnimo z vrednostnimi, ki se hranijo v spremenljivki »barvna_slika«. Z metodo »BitmapSource« pa prikažemo sliko na modulu »image1«. Prikaz barvne slike prožimo s potrditvenim poljem Barvni tok RGB.

```
void naprava_DepthFrameReady(object sender, DepthImageFrameReadyEventArgs e)
{
    if (checkBox2.IsChecked == true)
    {
        using (DepthImageFrame globinska_slika = e.OpenDepthImageFrame())
        {
            if (globinska_slika == null)
            {
                return;
            }
            byte[] pixels = GlobinskaSlika(globinska_slika);
            int sirinaslike = globinska_slika.Width * 4;
            image1.Source = BitmapSource.Create(globinska_slika.Width, globinska_slika.Height, 96, 96,
                PixelFormats.Bgr32, null, pixels, sirinaslike);
        }
    }
}
```

Programska koda 3.4: Funkcija za prikaz globinske slike.

Na podoben način deluje tudi funkcija »naprave_DepthFrameReady« (Koda 3.4), ki skrbi za globinski prikaz slike in zaznavanje oseb oziroma skeletov. Razlika med njima je v tem, da moramo pred prikazom slike na modulu

»image1« globinske slike dodatno obdelati s pomočjo funkcije »GlobinskaSlika« (Koda 3.5). Ta funkcija poskrbi, da iz globinske sličice izračunamo globino oz. oddaljenost od Kinecta in število oseb oziroma skeletov. S pomočjo funkcije obarvamo tudi območje osebe oziroma skeleta z določeno barvo. Glede na oddaljenost od naprave pa ustrezno določimo barvo območja s pomočjo odtenkov sivine. Prikaz globinske slike prožimo s potrditvenim poljem »Globinski tok CMOS«, barvo skeletov pa s potrditvenim poljem »Omogoči barvo oseb«.

```
private byte[] GlobinskaSlika(DepthImageFrame globinska_slika)
{
    //dobim neobdelane globinske podatke iz CMOS senzorja za vsak PIXEL Posebej
    //shranim jih v short tabelo
    short[] ng_podatki = new short[globinska_slika.PixelDataLength];

    // ng_podatki - neobdelaniglobinskipodatki - RAW
    globinska_slika.CopyPixelDataTo(ng_podatki);

    //*4 za razlicne barve
    Byte[] piksli = new byte[globinska_slika.Height * globinska_slika.Width * 4];

    const int Modra = 0;
    const int Zelena = 1;
    const int Rdeca = 2;

    //gremo skozi vse razdalje, ki jih uporablja Kinect
    //izberi Barvo RGB glede na razdaljo od Kinecta

    for (int inxglobina = 0, inxbarva = 0;
    inxglobina < ng_podatki.Length && inxbarva < piksli.Length;
    inxglobina++, inxbarva += 4)
    {
        //ID OSEBE
        int oseba = ng_podatki[inxglobina] & DepthImageFrame.PlayerIndexBitmask;
        //globina z bitshift // v vrednosti vraca v mm 1000 - 1m
        int globina = ng_podatki[inxglobina] >> DepthImageFrame.PlayerIndexBitmaskWidth;
        //Izracun oddtenka
        byte jakost = (byte)(255 - (255 * Math.Max(globina - minglobina, 0)/(razlikaglobina)));

        //monokromatski prikaz
        piksli[inxbarva + Modra] = jakost;
        piksli[inxbarva + Zelena] = jakost;
        piksli[inxbarva + Rdeca] = jakost;

        //razlicna barve za osebe
        if (checkBox4.IsChecked == true && checkBox2.IsChecked==true)
        {
            switch (oseba)
            {
                case 1:
                    piksli[inxbarva + Modra] = Colors.Yellow.B;
                    piksli[inxbarva + Zelena] = Colors.Yellow.G;
                    piksli[inxbarva + Rdeca] = Colors.Yellow.R;
                    break;
                case 2:
                    piksli[inxbarva + Modra] = Colors.Green.B;
                    piksli[inxbarva + Zelena] = Colors.Green.G;
```

```
                    piksli[inxbarva + Rdeca] = Colors.Green.R;
                    break;
                case 3:
                    piksli[inxbarva + Modra] = Colors.Blue.B;
                    piksli[inxbarva + Zelena] = Colors.Blue.G;
                    piksli[inxbarva + Rdeca] = Colors.Blue.R;
                    break;
                case 4:
                    piksli[inxbarva + Modra] = Colors.Red.B;
                    piksli[inxbarva + Zelena] = Colors.Red.G;
                    piksli[inxbarva + Rdeca] = Colors.Red.R;
                    break;
                case 5:
                    piksli[inxbarva + Modra] = Colors.Violet.B;
                    piksli[inxbarva + Zelena] = Colors.Violet.G;
                    piksli[inxbarva + Rdeca] = Colors.Violet.R;
                    break;
                case 6:
                    piksli[inxbarva + Modra] = Colors.Pink.B;
                    piksli[inxbarva + Zelena] = Colors.Pink.G;
                    piksli[inxbarva + Rdeca] = Colors.Pink.R;
                    break;
                case 7:
                    piksli[inxbarva + Modra] = Colors.Azure.B;
                    piksli[inxbarva + Zelena] = Colors.Azure.G;
                    piksli[inxbarva + Rdeca] = Colors.Azure.R;
                    break;
                default:
                    break;
            }
        }
    }
    return piksli;
```

Programska koda 3.5: Funkcija za pravilno obdelavo globinskega toka.

Obstaja pa tudi splošna funkcija »naprava_AllFramesReady« (Koda 3.6), ki lahko upravlja z vsemi tremi tokovi. V aplikaciji jo uporabljamo za upravljanje skeleta in klicanje funkcij za upravljanje aplikacij. Funkcijam posredujemo najdeni skelet.

```
void naprava_AllFramesReady(object sender, AllFramesReadyEventArgs e)
{
    skrijskelet();
    if (checkBox3.IsChecked == true)
        {
            Skeleton prvi = NajdiPrviSkelet(e);
            if (prvi == null)
            {
                return;
            }
        pokaziskelet();
        PostaviIzrisiSkelet(prvi, e);
            if (checkBox5.IsChecked == true)
            {
                UpravljalnikPPT(prvi);
            }
            if (checkBox6.IsChecked == true)
            {
                UpravljanikGOMPLAYER(prvi);
```

```
            }
        }
}
```

Programska koda 3.6: Funkcija, ki lahko upravlja vse tokove.

Skelet poiščemo s funkcijo »NajdiPrviSkelet« (Koda 3.7). Naprej napovemo, da bomo uporabljali skeletne slike – okvirje. Zatem sledi preverba ob izgubi slik – okvirjev. Informacije o skeletu prenesemo v polje skeletov s pomočjo funkcije »CopySkeletonDataTo«, S pomočjo stavka »select« – poizvedbe iz polja oziroma tabele »vsa_telesa« izberemo prvi skelet.

```
Skeleton NajdiPrviSkelet(AllFramesReadyEventArgs e)
{
    using (SkeletonFrame skelet = e.OpenSkeletonFrame())
    {
        if (skelet == null)
        {
            return null;
        }
        skelet.CopySkeletonDataTo(vsa_telesa);
        Skeleton prvi = (from s in vsa_telesa
                         where s.TrackingState == SkeletonTrackingState.Tracked
                         select s).FirstOrDefault();
        return prvi; // RABIM SAMO PRVEGA
    }
}
```

Programska koda 3.7: Funkcija, ki vrne prvo osebo oziroma skelet.

Prikaz skeletnih točk prožimo s potrditvenim poljem »Delo s skeletom«. Za upravljane določene aplikacije z gibi pa moramo imeti izbrano eno od potrditvenih polj – »PPT upravljalnik« ali »GOM upravljalnik«. Ti polji postaneta aktivni šele, ko je izbrano potrditveno polje »Delo s skeletom«. S pomočjo funkcije »PostaviIzrisiSkelet« (Koda 3.8) na pravilna mesta postavimo like oziroma točke skeleta. Pri tem moramo paziti, da točke umestimo na globinsko in barvno sliko. To naredimo s pomočjo funkcij »MapFromSkeletonPoint« in »MapToColorImagePoint«. Za dokončni izris pa poskrbi funkcija »PostaviTočke« (Koda 3.9), ki točke oziroma like postavi tudi na modul Canvas. Ob pritisku na Gumb »Nastavi na« se sproži funkcija (Koda 3.10), ki spremeni naklon Kinecta.

```
private void PostaviIzrisiSkelet(Skeleton prvi, AllFramesReadyEventArgs e)
{
    using (DepthImageFrame globinska_slika = e.OpenDepthImageFrame())
    {
        if (globinska_slika == null)
        {
            return;
        }
        pokaziskelet();
        // mapiramo sklepe na globinski sliko
        DepthImagePoint g_glava = globinska_slika.MapFromSkeletonPoint(prvi.Joints[JointType.Head].Position);
        DepthImagePoint g_levaroka =
                globinska_slika.MapFromSkeletonPoint(prvi.Joints[JointType.HandLeft].Position);
        DepthImagePoint g_desnaroka =
                globinska_slika.MapFromSkeletonPoint(prvi.Joints[JointType.HandRight].Position);
        //mapiramo sklepe na barvno sliko - potreben se format slike
        ColorImagePoint b_glava = globinska_slika.MapToColorImagePoint(g_glava.X, g_glava.Y,barvni_format);
        ColorImagePoint b_levaroka = globinska_slika.MapToColorImagePoint(g_levaroka.X, g_levaroka.Y,
                barvni_format);
        ColorImagePoint b_desnaroka = globinska_slika.MapToColorImagePoint(g_desnaroka.X, g_desnaroka.Y,
                barvni_format);

        PostaviTocke(rectangle1, b_glava);
        PostaviTocke(ellipse1, b_levaroka);
        PostaviTocke(ellipse2, b_desnaroka);
    }
}
```

Programska koda 3.8: Funkcija, ki izriše skeletne točke.

```
private void PostaviTocke(FrameworkElement lik, ColorImagePoint tocka_skeleta)
{
    Canvas.SetLeft(lik, tocka_skeleta.X - lik.Width);
    Canvas.SetTop(lik, tocka_skeleta.Y - lik.Height);
}
```

Programska koda 3.9: Funkcija, ki postavi like na določene točke.

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    button1.IsEnabled = false;
    if (naprava != null && naprava.IsRunning)
    {
        naprava.ElevationAngle = (int)slider1.Value;
        label1.Content = naprava.ElevationAngle.ToString();
        label2.Content = naprava.ElevationAngle.ToString();
    }
    System.Threading.Thread.Sleep(new TimeSpan(hours: 0, minutes: 0, seconds: 5)); // preprosta zascita pred
        obrabo motorcka
    button1.IsEnabled = true;
}
```

Programska koda 3.10: Funkcija za upravljane nagiba Kinecta.

Razvili smo tri funkcije, ki skrbijo za pretvorbo gibov v ukaze. Ukaze posredujejo aktivni aplikaciji. Vsaka funkcija upravlja s svojo osjo točke skeleta. To pomeni, da lahko na podlagi pomika rok ali drugega dela skeleta v smereh

x, y, z določimo neko kretnjo in za to kretnjo tudi ukaz, ki se posreduje aplikaciji. Funkcija »Procesor_Ukazov_Roke_OS_X« (Koda 3.11) spremlja gibe leve in desne roke v smeri osi x. Ob pravilnih gibih, ki sta definirana v pogojnih stavkih, izvede oziroma pošlje posredovana ukaza aktivni aplikaciji.

```
private void Procesor_Ukazov_Roke_OS_X(float sklep1, float sklep2, float sklep3, float frazdalja, string p1, string p2)
{
    if (sklep3 > (sklep1 + frazdalja))
    {
        if (!aktivnaleva)
        {
            System.Windows.Forms.SendKeys.SendWait(p1);

            aktivnaleva = true;
        }
    }
    else
    {
        aktivnaleva = false;
    }

    if (sklep2 < (sklep1 - frazdalja))
    {
        if (!aktivnadesna)
        {
            System.Windows.Forms.SendKeys.SendWait(p2);
            aktivnadesna = true;
        }
    }
    else
    {
        aktivnadesna = false;
    }
}
```

Programska koda 3.11: Funkcija za ukaze rok v smeri osi x.

Funkcija »Procesor_Ukazov_Roke_OS_Y« (Koda 3.12) spremlja gibe leve in desne roke v smeri osi y. Ob pravilnih gibih, ki sta definirana v pogojnih stavkih, izvede oziroma pošlje posredovana ukaza aktivni aplikaciji.

```
private void Procesor_Ukazov_Roke_OS_Y(float sklep1, float sklep2, float sklep3, float frazdalja, string p1, string p2)
{
    if ((sklep3 > (sklep1 + frazdalja)) && (sklep2 > (sklep1 + frazdalja)))
    {
        if (!dol)
        {
            System.Windows.Forms.SendKeys.SendWait(p1);

            dol = true;
        }
    }
    else
    {
        dol = false;
    }
    if ((sklep3 < (sklep1 - frazdalja)) && (sklep2 < (sklep1 - frazdalja)))
    {
        if (!gor)
```

```
            {
                System.Windows.Forms.SendKeys.SendWait(p2);
                gor = true;
            }
        }
        else
        {
            gor = false;
        }
    }
```

Programska koda 3.12: Funkcija za ukaze rok v smeri osi y.

Funkcija »UpravljanikPPT« (Koda 3.13) je namenjanja za upravljanje PowerPoint projekcije. Funkcija »UpravljanikGOMPLAYER« (Koda 3.14) je namenjanja preprostemu upravljanju predvajalnika medijev GOM Player.

```
    private void UpravljalnikPPT(Skeleton prvi)
    {
        var glava = prvi.Joints[JointType.Head];
        var cramena = prvi.Joints[JointType.ShoulderCenter];
        var ctelesa = prvi.Joints[JointType.HipCenter];
        var hrbtenica = prvi.Joints[JointType.Spine];
        var levaroka = prvi.Joints[JointType.HandLeft];
        var desnaroka = prvi.Joints[JointType.HandRight];

        Procesor_Ukazov_Roke_OS_Y(hrbtenica.Position.Y, levaroka.Position.Y, desnaroka.Position.Y, faktorrazdaljeX,
            "{F5}", "{ESC}");
        Procesor_Ukazov_Roke_OS_X(glava.Position.X, levaroka.Position.X, desnaroka.Position.X, faktorrazdaljeX,
            "{Right}", "{Left}");
    }
```

Programska koda 3.13: Upravljanik PPT.

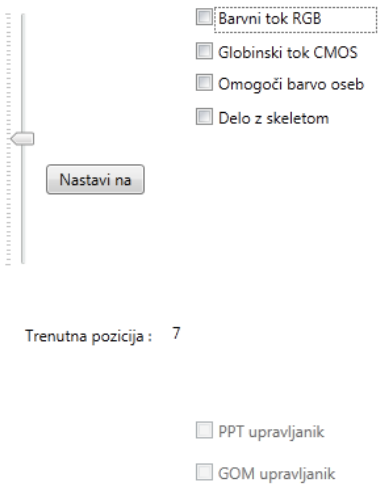
```
    private void UpravljanikGOMPLAYER(Skeleton prvi)
    {
        var glava = prvi.Joints[JointType.Head];
        var cramena = prvi.Joints[JointType.ShoulderCenter];
        var ctelesa = prvi.Joints[JointType.HipCenter];
        var hrbtenica = prvi.Joints[JointType.Spine];
        var levaroka = prvi.Joints[JointType.HandLeft];
        var desnaroka = prvi.Joints[JointType.HandRight];

        Procesor_Ukazov_Roke_OS_Y(hrbtenica.Position.Y, levaroka.Position.Y, desnaroka.Position.Y, faktorrazdaljeY,
            " ", "UP");
        Procesor_Ukazov_Roke_OS_X(glava.Position.X, levaroka.Position.X, desnaroka.Position.X, faktorrazdaljeX,
            "{PGDN}", "{PGUP}");

    }
```

Programska koda 3.14: Upravljanik GOMPLAYER.

Ker smo aplikacijo razvijali v SDK različici 1 spada pod licenco Microsoft Public License (Ms-PL).

3.2.3 Uporabniška navodila za uporabo aplikacije

Program Upravljalnik Kinect nam omogoča osnovno upravljanje Kinecta. Vse spremembe izvajamo preko ukaznega dela aplikacije. Naklon Kinecta upravljamo tako, da z drsnikom nastavimo želen naklon. Spremembo naklona sprožimo s pritiskom na gumb »Nastavi na«. Spreminjanje prikazanih tokov upravljamo s potrditvenimi polji. Če želimo imeti prikazano barvno sliko moramo izbrati polje »Barvni tok RGB«, če želimo globinsko sliko izberemo polje »Globinski tok CMOS«. S kodo smo onemogočili hkratno izbiro obeh tokov. Pri globinskem zajemu slike lahko s pomočjo polja »Omogoči barvo oseb« dodelimo ogrodju skeleta fiksno določeno barvo. To polje deluje le, če delamo z globinskim tokom.



Slika 3.7: Ukazni del aplikacije.

Ko želimo upravljati aplikacije pa moramo najprej izbrati polje »Delo s skeletom«, ki poskrbi, da bomo sledili delom telesa – v naši aplikaciji so to roke. Za lažji pregled se na mestu kjer sta roki in glava pojavijo manjši krogi. Ko je polje »Delo s skeletom« izbrano se nam omogočita polji za upravljane aplikacij. Upravljanje projekcije PowerPoint poteka tako, da izberemo potrditveno polje »PPT upravljalnik«. Projekcija mora teči v ospredju. Ukazi so fiksno določeni. Projekcijo začnemo z dvigom rok. Zamah desne roke v desno nam prestavi na naslednji stran. Zamah leve roke v levo pa na prejšnjo stran. Na podoben način deluje upravljalnik GOM predvajalnika. Sprememba je le ta, da dvig rok pomeni ukaz predvajaj ali pavza. Zamah desne roke v desno nam prestavi na naslednjo pesem ali film. Zamah leve roke v levo pa na prejšnjo pesem ali film.

3.2.4 Možne izboljšave aplikacije

Možnosti za izboljšavo je veliko. Prva izboljšava bi bila samodejno zaznavanje aktivne aplikacije in samodejna izbira ukazov zanjo. Ustvarili bi bazo aplikacij, ki hrani kretnje za ukaze trenutno aktivne aplikacije. Izvedli bi lahko tudi vmesnik za določevanje novih kretenj ukazom. V nadaljevanju bi lahko naredili vmesnik za glasovne ukaza.

Ena od izboljšav bi bila tudi temeljita sprememba uporabniškega vmesnika, ki je zdaj precej preprost. Z vpeljavo dodatnih funkcij bi ga nadgradili in prilagodili, da bi bil prijaznejši za uporabo. Popravili bi lahko tudi izgled samega programa.

Možna izboljšava aplikacije bi bila upravljanje tipkovnice in miške. S tem bi lahko upravljali Windows 7, Windows 7 Embedded ali prihajajoči Windows 8 s pomočjo Kinecta. To bi lahko realizirali z razvojem dveh modulov, ki bi skrbela samo za ukaze, povezane z miško in tipkovnico. Okolje za upravljane s tipkovnico in miško bi lahko postavili v 3D prostor v smislu virtualnega upravljalnika. S tem pa bi izkoriščali tudi novejše 3D prikazovalnike slike.

Poglavje 4

Sklepne ugotovitve

V prvem delu diplomske naloge smo se podrobneje seznanili s senzorjem Kinect in z različnimi napravami, ki so podobne senzorju Kinect. Izvedli smo primerjavo naprav s Kinectom. Opisali smo različne SDK-je in knjižnice, ki jih lahko uporabljamo s senzorjem Kinect. Navedli smo tudi nekaj aktualno zanimivih projektov realiziranih s Kinectom.

V drugem delu diplomske naloge smo predstavili lastno razvito aplikacijo, ki uporablja senzor Kinect. Razvita aplikacija je preprosta in ima še veliko omejitev, predvsem v smislu univerzalnosti in prilagodljivosti. Trenutno ima vključena modula za vodenje PowerPoint projekcije in preprosto upravljanje predvajalnika medijev GOM predvajalnika. Aplikacijo lahko precej nadgradimo, ker obstaja še veliko možnosti, ki bi jih lahko vključili.

Pri razvoju aplikacije smo se seznanili z uporabo Kinectovega SDK-ja, dela s Kinectovimi senzorji, sistematičnim postopkom razvoja aplikacije in programskim jezikom C#.

Za popolni izkoristek novih različic SDK-jev je potreben nakup PC verzije Kinecta. Microsoftov SDK nam omogoča hiter začetek in razvoj aplikacij namenjenih za Kinect. Edina slabost razvoja aplikacij s temi SDK-ji je, da jih lahko uporabljamo samo na operacijskih sistemih Windows 7, Windows 7 Embedded in prihajajočem Windows 8. Če želimo resnično, da bodo aplikacije delovale na različnih operacijskih sistemih, je potreben razvoj aplikacij s pomočjo OpenNi ali OpenKinect/libfreenect ogrodja.

Slike

Tabele

Programska koda

Literatura

[1] (2012) Active pixel sensor – Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/Active_pixel_sensor

[2] (2012) Augmented Reality(AR) Sandbox. Dostopno na:
http://www.idav.ucdavis.edu/~okreylos/ResDev/SARndbox/
index.html

[3] (2012) Augmented Reality(AR) Sandbox - slike. Dostopno na:
http://www.idav.ucdavis.edu/~okreylos/ResDev/SARndbox/
Pictures.html

[4] (2012) Augmented Reality(AR) Sandbox - filma. Dostopno na:
http://www.idav.ucdavis.edu/~okreylos/ResDev/SARndbox/
Movies.html

[5] (2012) Bayer filter – Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/Bayer_filter

[6] (2012) Binocular vision – Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/Binocular_vision

[7] (2012) CL NUI Platform. Dostopno na:
http://codelaboratories.com/kb/nui

[8] (2012) Code Laboratories. Dostopno na:
http://codelaboratories.com/products/

[9] (2012) Coding4Fun Kinect. Dostopno na:
http://channel9.msdn.com/coding4fun/kinect#tab_sortBy_recent

[10] (2012) Hardware info. Dostopno na:
http://openkinect.org/wiki/Hardware_info

[11] (2012) Kinect for Windows. Dostopno na:
http://www.microsoft.com/en-us/kinectforwindows/

[12] (2012) Kinect for Windows SDK v1.5. Dostopno na:
http://www.microsoft.com/en-us/download/details.aspx?id=29866

[13] (2012) Kinect for Windows Developer Toolkit v1.5.1. Dostopno na:
http://www.microsoft.com/en-us/download/details.aspx?id=30140

[14] (2012) Kinect for Windows Language Packs v1.5.0. Dostopno na:
http://www.microsoft.com/en-us/download/details.aspx?id=29864

[15] (2012) Kinect for Windows SDK and Toolkit – v1.5 Release Notes. Dostopno na:
http://msdn.microsoft.com/en-us/library/jj131036

[16] (2012) Kinect Hacking. Dostopno na:
http://idav.ucdavis.edu/~okreylos/ResDev/Kinect/index.html

[17] (2012) Kinect – Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/Kinect

[18] (2012) Kinect Dashhaks. Dostopno na:
http://kinect.dashhacks.com/

[19] (2012) Spletna stran KinectHacks. Dostopno na:
http://www.kinecthacks.com/

[20] (2012) Kinect Pattern Uncovered. Dostopno na:
`http://azttm.wordpress.com/`

[21] (2012) Microsoft Kinect Teardown. Dostopno na:
`http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/1`

[22] (2012) Microsoft MSDNA - About YUV Video. Dostopno na:
`http://msdn.microsoft.com/en-us/library/windows/desktop/bb530104\%28v=vs.85\%29.aspX`

[23] (2012) OpenKinect. Dostopno na:
`http://openkinect.org/wiki/Main_Page`

[24] (2012) Orodje Vrui VR. Dostopno na:
`http://idav.ucdavis.edu/~okreylos/ResDev/Vrui/index.html`

[25] (2012) PlayStation 3 – Wikipedia. Dostopno na:
`http://en.wikipedia.org/wiki/PlayStation_3`

[26] (2012) PlayStation Eye – Wikipedia. Dostopno na:
`http://en.wikipedia.org/wiki/PlayStation_Eye`

[27] (2012) PlayStation EyeToy – Wikipedia. Dostopno na:
`http://en.wikipedia.org/wiki/EyeToy`

[28] (2012) PlayStation Move – Wikipedia. Dostopno na:
`http://en.wikipedia.org/wiki/PlayStation_Move`

[29] (2012) Project Icarus. Dostopno na:
`http://www.kinecthacks.com/project-icarus/`

[30] (2012) Project Icarus – posnetek prvega testa. Dostopno na:
`http://www.youtube.com/watch?v=RDLWG5-KeVg&feature=player_embedded`

[31] (2012) Project Icarus – posnetek drugega testa. Dostopno na:
http://www.youtube.com/watch?v=PlHUOHr4WxA

[32] (2012) RGB color model – Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/RGB_color_model

[33] (2012) Stereo camera – Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/Stereo_camera

[34] (2012) Structured-light 3D scanner – Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/Structured-light_3D_scanner

[35] (2012) Ustanova Renato Archer Center for Information Technogy - CTI Dostopno na:
http://www.cti.gov.br/english/

[36] (2012) Wii Remote – Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/Wii_Remote

[37] (2012) Wii Balance Board – Wikipedia. Dostopno na:
http://en.wikipedia.org/wiki/Wii_Balance_Board

[38] (2011) Asus Xtion Pro. Dostopno na:
http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO/

[39] (2011) Asus Xtion Pro Live. Dostopno na:
http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO_LIVE/

[40] (2011) Interaktivni češki peskovnik - posnetek. Dostopno na:
http://www.youtube.com/watch?v=8p7YVqyudiE

[41] (2011) Kinect Virtual Reality with 3D TV - posnetek. Dostopno na:
http://www.youtube.com/watch?v=2MX1RinEXUM&feature=related

[42] (2011) Kinect Virtual Reality with 3D TV - o avtorju. Dostopno na:
http://csc.lsu.edu/~kooima/index.html

[43] (2011) Kinect Virtual Reality with 3D TV - o Electro VR. Dostopno na:
`http://csc.lsu.edu/~kooima/electro/electro.html`

[44] (2011) OpenNi. Dostopno na:
`http://www.openni.org/`

[45] (2011) AscTec Pelican. Dostopno na:
`http://www.asctec.de/asctec-pelican-3/`

[46] (2011) Projekt InVesalius. Dostopno na:
`http://svn.softwarepublico.gov.br/trac/invesalius`

[47] (2011) Projekt Mimicry. Dostopno na:
`http://www.monobanda.nl/index_en.html`

[48] (2011) Projekt NAVI. Dostopno na:
`http://hci.uni-konstanz.de/blog/2011/03/15/navi/?lang=en`

[49] (2011) Projekt NAVI - posnetek o testu sistema. Dostopno na:
`http://www.youtube.com/watch?v=l6QY-eb6NoQ`

[50] (2011) Projekt NAVI - koda aplikacije. Dostopno na:
`http://navi.codeplex.com/`

[51] (2011) Projekt Quadrotor + Kinect – uradna stran. Dostopno na:
`http://hybrid.eecs.berkeley.edu/~bouffard/kinect.html/`

[52] (2011) ROS gonilniki za Kinect RGB-D kamero. Dostopno na:
`http://www.ros.org/wiki/kinect_node/`

[53] (2011) ROS osnovna programska orodja za upravljanje helikopterjev STARMAC. Dostopno na:
`http://www.ros.org/wiki/starmac-ros-pkg`

[54] (2011) Vmesnik za upravljane posnetkov MRI - film Dostopno na:
`http://www.youtube.com/watch?v=yLqSY07tS64`

[55] (2011) Xbox Kinect in the hospital operating room - posnetek. Dostopno na:
http://www.youtube.com/watch?v=f5Ep3oqicVU

[56] (2011) Xbox useful to surgeons - izjava bolnišnice. Dostopno na:
http://sunnyview.sunnybrook.ca/2011/03/
fun-and-games-in-or.html

[57] (2011) Xbox useful to surgeons - izjave na spletnem dnevniku. Dostopno na:
http://sunnybrook.ca/media/item.asp?c=1&i=616&page=

[58] (2010) Kinect and Wiimote and Nanotech Construction Kit. Dostopno na:
http://www.youtube.com/watch?v=8fZJoKRjJBg

[59] (2010) Technical description of Kinect calibration. Dostopno na:
http://www.ros.org/wiki/kinect_calibration/technical#
Kinect_operation

[60] (2010) Vzorec Kinectove IR kamere. Dostopno na:
http://www.futurepicture.org/?p=129