UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Sebastjan Burnar

# Upravljanje računalnika s senzorjem Kinect

DIPLOMSKO DELO
VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Mentor: viš. pred. dr. Alenka Kavčič

Ljubljana, 2012

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: 00290/2012
Datum: 13.04.2012

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

Kandidat: **SEBASTJAN BURNAR**

Naslov: **UPRAVLJANJE RAČUNALNIKA S SENZORJEM KINECT**
**COMPUTER INTERACTION USING KINECT SENSOR**

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V diplomski nalogi preučite možnosti uporabe in integracije senzorja Kinect za nadzor in upravljanje računalnika oz. uporabniških aplikacij. Senzor Kinect lahko kot brezdotični vmesnik nadomesti klasične vmesnike za interakcijo človeka z računalnikom, pri tem pa lahko interakcijo nadgradi tudi z uporabo gest. Zato posebno pozornost usmerite v prepoznavanje gest uporabnika, pri čemer naj bo uporabniku omogočeno tudi definiranje lastnih gest za upravljanje računalnika oz. aplikacije.
V okviru diplomske naloge izdelajte aplikacijo, ki omogoča brezdotično interakcijo uporabnika s ustrezno prilagojenimi spletnimi stranmi. Na koncu ocenite prednosti realiziranega brezdotičnega uporabniškega vmesnika pred klasičnimi vmesniki ter podajte kritično presojo njegovega delovanja.

Mentor:
Alenka Kavčič
prof. dr. Alenka Kavčič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Sebastjan Burnar, z vpisno številko 63040196, sem avtor diplomskega dela z naslovom:

Upravljanje računalnika s senzorjem Kinect

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Alenke Kavčič,

- 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 15. oktobra 2012 Podpis avtorja:

# Zahvala

Zahvaljujem se viš. pred. dr. Alenka Kavčič za mentorstvo, potrpežljivost, nasvete, pomoč ter vodenje pri pisanju diplomskega dela.

Zahvaljujem se podjetju Neolab, sestri, mami, bratrancem in prijateljem za podporo in spodbudne besede.

# Kazalo

**Povzetek**

**Abstract**

**1 Uvod 1**

**2 Kaj je Kinect 3**

2.1 Kamera RGB ... 3

2.2 Globinski senzor ... 4

2.3 Štirje mikrofoni ... 5

2.4 Motoriziran podstavek ... 5

2.5 Kako senzor Kinect deluje? ... 5

2.6 Alternativa senzorju Kinect ... 6

**3 Uporabljene tehnologije 7**

3.1 Microsoft Visual C# 2010 Express ... 7

3.2 Kinect SDK 1.5 ... 8

3.2.1 Objekt KinectSensor ... 8

3.2.2 Slikovni tok ... 9

3.3 Kinect Toolkit ... 11

3.3.1 Toolkit Browser ... 12

3.3.2 Kinect Studio ... 12

**4 Aplikacija geste 15**

4.1 Kaj je gesta? ... 15

4.2 Kako zaznamo gesto v aplikaciji ... 15

4.3 Knjižnica KinectGestureLib ... 16

4.4 Primer zaznavanja geste krog ... 17

4.5 Opis aplikacije geste ... 18

**5 Brezdotični brskalnik** **21**

**6 Sklep** **25**

6.1 Predlogi za nadaljnje delo ..... 25

**Kazalo slik**

**Kazalo izvorne kode**

**Literatura**

# Seznam uporabljenih kratic in simbolov

**3D** – tridimenzionalno

**COM** - (Component Object Model) komponentni objektni model

**HTML** - HyperText Markup Language

**IR** - (Infrared) infrardeča svetloba

**JavaScript** - (JS) skriptni jezik

**JSON** - (JavaScript Object Notation) zapis objekta v skriptnem jeziku JavaScript

**Kinect** - senzor za Microsoftovo konzolo Xbox 360

**RGB** - (Red, Green, Blue) je barvni model, ki ga sestavljajo rdeča, zelena in modra barva

**SDK** - (Software Develpopemnt Kit) orodja za razvoj aplikacij

**Toolkit** - sklop orodij za razvoj aplikacij

**WPF** - Windows Presentation Foundation

**XML** – (Extensible Markup Language) množica pravil za kodiranje dokumentov

# Povzetek

Cilj diplomske naloge je predstavitev senzorja Kinect in preučitev možnosti uporabe z računalnikom. V prvem delu diplomska naloga opisuje senzor Kinect, njegove komponente in delovanje. Predstavljeno je orodje Microsoft Visual C# 2010 Express in paket Kinect SDK, ki skupaj omogočata razvoj aplikacij za senzor Kinect, ter paket Kinect Toolkit, kjer so predstavljeni primeri uporabe in dokumentacija.

V drugem delu diplomske naloge sta opisani dve aplikaciji. Prva aplikacija je aplikacija geste za zaznavanje gest. Opisan je postopek zaznavanja gest in podan primer zaznavanja geste krog. Aplikacija geste za realizacijo in detekcijo gest uporablja knjižnico KinectGestureLib, ki smo jo v ta namen razvili v okviru diplomske naloge.

Druga aplikacija je brezdotični brskalnik. Opisane so posamezne komponente, ki sestavljajo aplikacijo, in predstavljena je ena od rešitev, kako lahko senzor Kinect nadomesti napravo, kot je miška.

**Ključne besede**: senzor, Kinect, brezdotični vmesnik, Microsoft, SDK, Visaul Visual C# 2010 Express, gesta, aplikacija, RGB, IR, brskalnik.

# Abstract

The thesis presents the Kinect sensor and explores the possibilities of using it with a computer. The first part of the thesis describes the Kinect sensor, its features and operation. Presented is the Microsoft Visual C # 2010 Express tool, Kinect SDK package which can be used for developing applications for the Kinect sensor, and Kinect Toolkit package, which offers examples and documentation.

In the second part the thesis describes two applications. The first application is an application gesture for detecting gestures. The thesis outlines the procedure for detecting gestures and it describes an example of circle gesture detection. Application gesture uses the library KinectGestureLib for the realization and gesture detection.

The second application is a no-touch browser. The thesis describes the individual components that make up the application, and displays one of the solutions, in which the Kinect sensor can replace a different device, such as a mouse.



**Keywords**: sensor, Kinect, no-touch interface, Microsoft, SDK, Visaul Visual C# 2010 Express, gesture, application, RGB, IR, browser.

# 1 Uvod

Interakcija z elektronskimi napravami poteka preko vhodnih naprav, kot so tipkovnica, miška, grafične tablice, igralni ploščki itd. Večina vhodnih naprav uporablja za pošiljanje ukazov gumbe, pri katerih smo zelo omejeni, saj vsak gumb opravlja samo eno funkcijo - pritisk na gumb pošlje napravi informacijo, da je bil pritisnjen. Ekrani, občutljivi na dotik, so odpravili nepotrebne gumbe in omogočili izredno široko paleto novih načinov podajanja ukazov z dotikom. Senzor Kinect pa lahko uporabimo kot brezdotični vmesnik. Slednji odpravlja fizično interakcijo z napravo, kot so gumbi ali na dotik občutljive površine, kjer uporabnik napravi podaja ukaze z gibi telesa.

Predhodne naprave, ki so omogočale brezdotični vmesnik, so bile navadne kamere, ki so s pomočjo algoritmov prepoznavale objekte in ljudi na sliki. Senzor Kinect omogoča boljše prepoznavanje ljudi in objektov v prostoru kot navadne kamere. Omenjeni senzor za zaznavanje uporablja posebno tehnologijo, ki s pomočjo dveh kamer omogoča zajemanje prostora v 3D. Zajeti 3D prostor je predstavljen kot slika. Slednjo sestavljajo slikovne pike, ki vsebujejo podatek o globini, zato zajeti 3D sliki pravimo globinska slika. To globinsko sliko lahko uporabimo za skeniranje 3D objektov, ki jih potem uvozimo v modelarske programe za nadaljnjo obdelavo.

V zadnjih letih je bilo opaziti povečano število televizijskih in drugih ekranov v izložbah in ostalih lokacijah. Z relativno nizkimi stroški senzorja Kinect je mogoče tovrstno oglaševanje nadgraditi, kjer imajo uporabniki možnost interakcije, kar bi verjetno vplivalo na večjo učinkovitost omenjenega oglaševanja. Primerna uporaba senzorja Kinect bi bila v bolnišnici, kjer bi z uporabo brezdotičnega vmesnika zmanjšali prenos bakterij. Kirurg bi lahko med posegom z gestami na ekranu povečal ali zmanjšal sliko, spremenil informacijo itd. Zato smo v okviru diplomske naloge razvili dve aplikaciji, ki prikazujeta možnost uporabe senzorja Kinect. V prvi aplikaciji smo predstavili zaznavanje in urejanje gest. Uporabnik bi s pomočjo preprostih gest izbiral med poljubno vsebino na ekranu. V drugi aplikaciji pa smo predstavili brezdotični brskalnik, ki prikazuje prilagojene spletnih strani in z uporabo senzorja Kinect simuliramo računalniško miško.

# 2 Kaj je Kinect

Senzor Kinect [10] je igralni pripomoček za Microsoftovo konzolo Xbox 360. Njegov namen je igranje iger, pri katerih igralec namesto igralnega ploščka podaja ukaze z gibi telesa in glasom. Izdelalo ga je hčerinsko podjetje Rare družbe Microsoft Game Studios, ki je v lasti podjetja Microsoft. Ker omogoča globinsko zaznavanje, je poleg igranja iger vse bolj priljubljen pri razvijalcih naravnega uporabniškega vmesnika, robotike, računalniškega vida in, kot smo že omenili v uvodu, tridimenzionalnega zajemanja slik.

V podolgovatem ohišju ima kamero RGB, infrardeči projektor, enobarvno kamero oziroma tipalo in štiri mikrofone, kot je prikazano na sliki 1. V ohišju je tudi ventilator za hlajenje infrardečega projektorja. Podatke pošilja preko kabla USB.


Slika 1: Senzor Kinect.

## 2.1 Kamera RGB

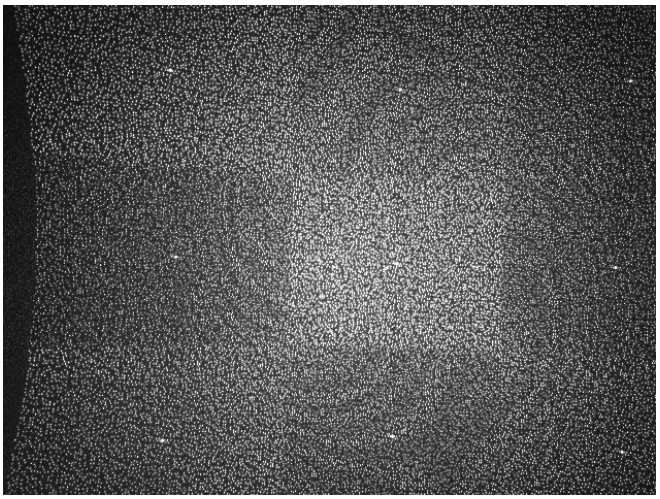
Kamera RGB zajema slike s hitrostjo 30 slik na sekundo v resoluciji 640 x 512 slikovnih točk, omogoča pa tudi opcijo zajemanja slik v resoluciji 1280 x 1024 slikovnih točk z manjšo hitrostjo, in sicer od 10 do 15 slik na sekundo. Kamera RGB ima dober nabor funkcij, kot so avtomatsko nastavljanje beline, avtomatska osvetlitev, zmanjšanje utripanja, nasičenost barv in popravek barvnega tona slike. Ena od prednosti kamere je tudi prepoznavanje obrazov, ki ga uporablja Xbox 360 za prijavo v sistem.

## 2.2 Globinski senzor

Infrardeči projektor in senzor infrardeče svetlobe CMOS (Complimentary metal-oxide-semiconductor), ki sta prikazana na sliki 2, skupaj sestavljata globinski senzor. Slednjega so razvili pri podjetju PrimeSense. Projektor in senzor sta narazen približno 7,5 centimetrov in skupaj tvorita globinsko sliko.

Slika 2: Infrardeči projektor in senzor.

Infrardeči projektor projicira vnaprej znan vzorec svetlečih pik. Tega sestavljajo pike, ki so razdeljene v devet polj, v vsakem izmed teh pa je na sredini točka, ki je svetlejša od ostalih. Infrardečo projekcijo vzorca svetlečih pik prikazuje slika 3, na kateri so dobro vidne tudi svetlejše sredinske točke vseh devetih polj.



Slika 3: Infrardeča projekcija projektorja.

Globina se izračuna s pomočjo triangulacije, kjer senzor Kinect primerja znani vzorec proti vzorcu, ki ga projicira projektor. Za to je zadolžen čip, ki zna v realnem času izračunati zamik pik in generirati trideset 11-bitnih slik na sekundo.

## 2.3 Štirje mikrofoni

Senzor Kinect ima štiri navzdol obrnjene mikrofone - eden je na levi in trije na desni strani. To je tudi razlog, zakaj je senzor Kinect tako širok. Taka postavitev daje najboljše rezultate za zaznavanje zvoka. Mikrofoni so zmožni prepoznavanja govora, lociranja izvora govora in redukcije šuma, ki je prisoten v prostoru.

## 2.4 Motoriziran podstavek

Motoriziran podstavek lahko premika senzor Kinect vertikalno za 43 stopinj, horizontalno za maksimalno 57 stopinj, možen pa je tudi nagib gor in dol za 27 stopinj. Premikanje omogoča avtomatsko prilagajanje igralcem v prostoru.

## 2.5 Kako senzor Kinect deluje?

Ena bistvenih komponent poleg zgoraj omenjenih senzorjev je programska plast, ki posreduje podatke, kaj zaznava strojna oprema . Za začetek so izdelovalci Kinecta zbrali ogromno število podatkov o gibanju ljudi, slednje pa so uporabili za strojno učenje algoritma, ki ga je razvil raziskovalec Jamie Shotton pri MSR Cambridgeu v Angliji [9]. Razvijalcem je tako uspelo razdeliti podatke na modele, ki predstavljajo ljudi različnih starosti, telesnih tipov, spola in oblačil. Z zbranimi podatki so sistem naučili, da uporabi pravi model in da za posameznega igralca ustvari skeletno postavitev in poskrbi za pravilno postavitev sklepov za posameznega igralca [5, 6].

Poleg hitrega prepoznavanja prostora in igralcev zna sistem prepoznati igralce po obrazu. Če se igralci prekrivajo ali stojijo za objektom, zna sistem oceniti, v kakšni poziciji ali gibanju so njihovi prekriti deli telesa med igro.

## 2.6 Alternativa senzorju Kinect

Podjetje ASUS je ustvarilo senzorje gibanja Xtion [15], Xtion PRO [16] in Xtion PRO LIVE [17]. Gre za senzorje gibanja, ki jih vidimo na sliki 4. Vse tri naprave imajo vgrajen projektor infrardeče svetlobe in infrardečo tipalo, ki skupaj tvorita globinsko sliko. Zajem globinske slike je med 0,8 do 3,5 metrov. Prenos podatkov in napajanje poteka preko kabla USB. Xtion PRO LIVE ima tudi dva mikrofona za zajem zvoka. Senzorji Xtion so zanimivi predvsem zato, ker za njih poleg operacijskega sistema Windows 7 lahko razvijamo še operacijski sistem Linux Ubuntu, Windows XP, Windows Visto in Android. Pri senzorju Kinect smo omejeni na izdelovanje aplikacij za operacijski sistem Windows 7 in Windows 8.


Slika 4: Senzorji gibanja Xtion, Xtion PRO in Ktion PRO LIVE.

# 3 Uporabljene tehnologije

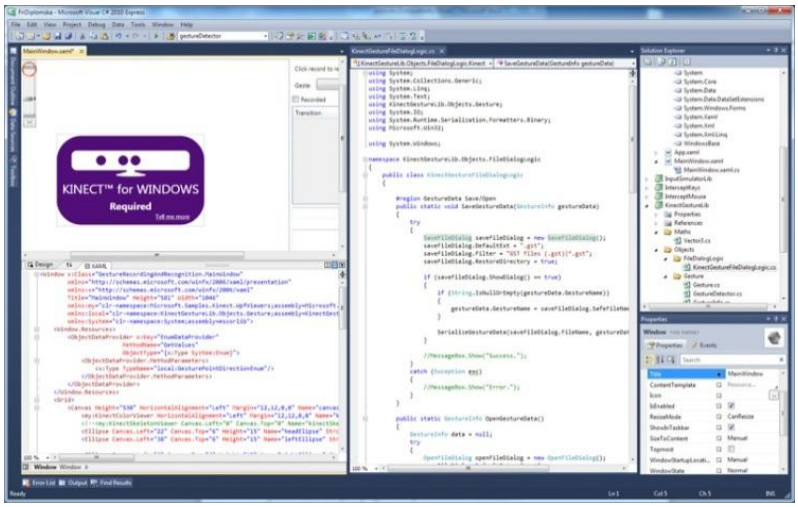
Aplikacijo Kinect smo razvijali na operacijskem sistemu Windows 7. Za razvoj smo uporabili:

- zastonjski urejevalnik kode Microsoft Visual C# 2010 Express,
- paket Kinect SDK 1.5,
- paket Kinect Toolkit 1.5.2.

## 3.1 Microsoft Visual C# 2010 Express

Microsoft Visual C# 2010 Express je orodje za urejanje, razhroščevanje in prevajanje kode, ki ga vidimo na sliki 5. Spada v zbirko Microsoftovih brezplačnih orodij in je namenjeno predvsem študentom začetnikom oziroma vsem tistim, ki se zanimajo za razvoj grafičnih in konzolnih aplikacij za operacijski sistem Windows.

Vsa orodja uporabljajo platformo .NET Framework [1], ki obsega zbirko raznovrstnih knjižnic, zaradi česar je del aplikacije lahko napisan v drugačnem programskem jeziku. Vse .NET framework aplikacije tečejo na virtualni komponenti CLR (Common Language Runtime) [2]. Komponenta skrbi za upravljanje s spominom, varnost, za boljšo podporo in večnitnost. Programski jezik C# [12] je sodoben, objektno usmerjen programski jezik, ki so ga razvili pri podjetju Microsoft. Sintaksa jezika je podobna jeziku Java [7].




Slika 5: Microsoft Visual C# 2010 Express.

## 3.2 Kinect SDK 1.5

Paket Kinect SDK [4] vsebuje gonilnike za uporabo senzorja Kinect z operacijskim sistemom Windows 7. Vsebuje tudi programski uporabniški vmesnik, ki omogoča dostop do tokov globinskega senzorja, RGB kamere, štirih mikrofonov in motoriziranega podstavka. Poleg dostopa do posameznih strojnih komponent imamo dostop do skeletne slike enega ali dveh uporabnikov. Gre za dostop do naprednega zaznavanja zvoka z redukcijo šuma, izničevanjem odmeva, lociranjem izvora zvoka in s prepoznavanjem govora.

Z verzijo Kinect SDK 1.5, ki je bila izdana 21. maja 2012, so izboljšali zaznavanje skeletne slike, ki je sedaj hitrejše in natančnejše. Dodali so sedeče sledenje - sistem tako sledi rokam, glavi in vratu uporabnika. Nova opcija »Near mode« omogoča, da globinski senzor vidi objekte od blizu že od 40 cm dalje. Izboljšana je tudi informacija poglobljenih vrednosti izven območja zaznave in sinhronizacija med kartiranjem barvne in globinske slike.

### 3.2.1 Objekt KinectSensor

Najpomembnejši objekt v paketu Kinect SDK je KinectSensor, preko katerega dostopamo do funkcionalnosti in tokov, ki nam jih ponuja senzor Kinect. Na računalnik imamo lahko povezanih več senzorjev Kinect. V Programski kodi 1 je primer kode, kako zaznamo prvi senzor Kinect, ki je priklopljen na računalnik.

```
KinectSensor kinectSensor;

// Find first sensor that is connected.
foreach (KinectSensor sensor in KinectSensor.KinectSensors)
{
        if (sensor.Status == KinectStatus.Connected)
        {
                this.kinectSensor = sensor;
        break;
        }
}
```

Programska koda 1: Poišče prvi senzor, ki je povezan na računalnik.

### 3.2.2 Slikovni tok

Slikovni tok nam omogoča dostop do podatkov, ki jih pošilja senzor Kinect računalniku preko kabla USB. Barvne tokove moramo pred zagonom senzorja Kinect omogočiti. To storimo s klicem metode »Enable()«, ki lahko prejme parameter, preko katerega nastavimo format zajemanja. V objektu KinectSensor imamo dostop do treh glavnih tokov. To so:

- barvni tok (ColorStream),
- globinski tok (DepthStream),
- skeletni tok (SkeletonStream).

#### 3.2.2.1 Barvni tok

Barvni tok nam omogoča dostop do barvne slike, ki jo pošilja RGB kamera. Za zajem slikovnega toka lahko izberemo enega od petih parametrov:

- ColorImageFormat.RawYuvResolution640x480Fps15,
- ColorImageFormat.RgbResolution1280x960Fps12,
- ColorImageFormat.RgbResolution640x480Fps30,
- ColorImageFormat.Undefined,
- ColorImageFormat.YuvResolution640x480Fps15.

#### 3.2.2.2 Globinski tok

Globinski tok nam pošilja podatke, ki jih posreduje globinski senzor. Tovrstna slika nam prikaže 3D prostor , kot ga vidi v svojem vidnem polju senzor Kinect. Vsaka slikovna točka predstavlja točko v prostoru. Globinsko sliko lahko predstavimo z barvami, kot lahko vidimo na sliki 6. Rdeče obarvane točke predstavljajo bližino, rumene oddaljenost. Točke, ki jih senzor Kinect ni zaznal, so obarvane črno.

Slika 1: Globinska slika.

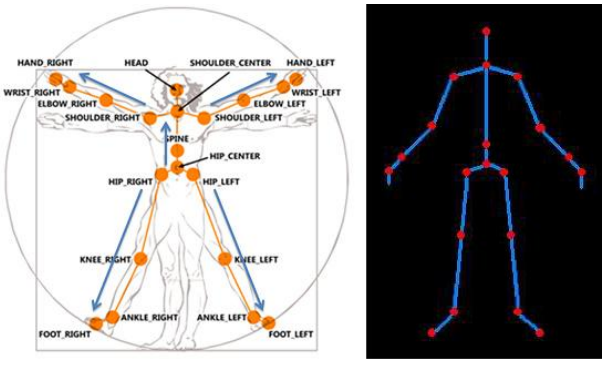
Za zajem globinskega toka lahko izberemo enega od štirih parametrov:

- DepthImageFormat.Resolution320x240Fps30,
- DepthImageFormat.Resolution640x480Fps30,
- DepthImageFormat.Resolution80x60Fps30,
- DepthImageFormat.Undefined.

### 3.2.2.3 Skeletni tok

Skeletni tok je podatkovni tok, ki nam omogoča prepoznavanje pozicije največ šestih uporabnikov. Poleg pozicije lahko sledi dvema uporabnikoma s kar 20 točkami sklepov v 3D prostoru. Točke sklepov so prikazane na sliki 7. Pozicija uporabnika in vsi podatki o sklepih so shranjeni v objektu Skeleton. Parametri objekta Skeleton so:

- SkeletonTrackingState - je status, ki nam pove, če senzor Kinect sledi uporabniku (Tracked/NotTracked). Sledi lahko dvema uporabnikoma hkrati.
- Joints - zbirka točk, ki predstavlja posamezen sklep uporabnika v 3D prostoru. Zbirka je dostopna samo, če senzor Kinect sledi uporabniku (Tracked).
- TrackingId - služi za identifikacijo posameznega skeleta uporabnika.
- Position - globalna pozicija uporabnika, sledi lahko do šestim uporabnikom hkrati.
- ClippedEdges - označuje, ali je del telesa izven vidnega polja senzorja Kinect.




Slika 2: Točke sklepov.

Senzor Kinect nima dovolj ločljivosti za zagotovitev točnosti sledenja uporabnika v daljšem časovnem obdobju. Ta problem se kaže pri podatkih, predvsem pri tistih o sklepih, kjer se zdi kot da posamezen sklep vibrira okrog svoje pozicije. Zato imamo na voljo možnost izravnavanja oziroma glajenja teh vrednosti, predvsem pozicije sklepov (Joints). Možnost

glajenja nam zagotovi boljšo uporabniško izkušnjo. To izravnavanje nam omogoča objekt »TransformSmoothParameters«, ki ga lahko podamo kot parameter pri zajemanju skeletnega toka .Primer inicializacije nam predstavlja programska koda 2. Parametri »TransformSmoothParameters« objekta so:

- Smoothing - nastavi parametre glajenja. Nižja vrednost povzroči manjše glajenje, zato so podatki bližje surovim podatkom. Višja vrednost povzroči večje glajenje, pri čemer se poveča latenca.
- Correction - določa višino popravka. Višja vrednost se popravlja hitreje, nižja vrednost počasneje, slednja pa se tako zdi bolj gladka.
- Prediction - predvideva naslednje okvirje slik in s tem blaži prekoračitev vrednosti.
- JitterRadius - določa, za koliko se zmanjša tresenje iz neobdelanih podatkov.
- MaxDeviationRadius - določa največji obseg odstopanja od surovih podatkov.

```
TransformSmoothParameters parameters = new TransformSmoothParameters();
parameters.Smoothing = 0.5f;
parameters.Correction = 0.3f;
parameters.Prediction = 0.4f;
parameters.JitterRadius = 1.0f;
parameters.MaxDeviationRadius = 0.5f;

kinectSenzor.SkeletonStream.Enable(parameters);
```

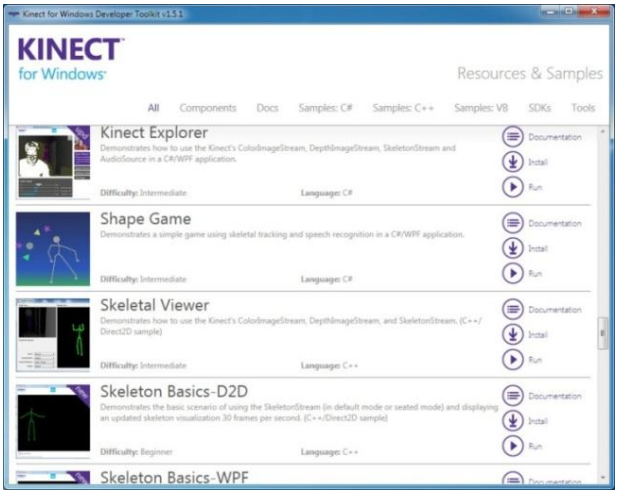
Programska koda 2: Objekt TransformSmoothParameters s parametri za glajenje.

## 3.3 Kinect Toolkit

Paket Kinect Toolkit [4] vsebuje primere uporabe z izvorno kodo in dokumentacijo. Vsebuje tudi dve nadvse priročni orodji, in sicer Toolkit Browser in Kinect Studio.

### 3.3.1 Toolkit Browser

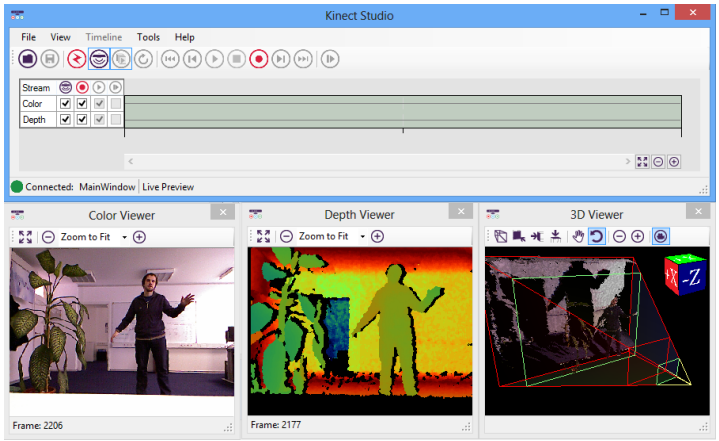
Toolkit Browser je aplikacija, kjer so predstavljeni vsi primeri uporabe v prijazni navigaciji in s filtrom. Kot vidimo na sliki 8, so poleg naslova, slike in opisa vsakega primera prisotni tudi gumbi do dokumentacije, gumb za namestitev in gumb za zagon. Primeri uporabe tako združujejo prikaz skeleta, sledenje obrazu, zelen zaslon, prikaz globinske slike, zaznavanje zvoka in govora, igre itd.




Slika 3: Aplikacija Toolkit Browser.

### 3.3.2 Kinect Studio

Kinect Studio [14] je orodje za snemanje barvne in globinske slike tokov, ki jih pošilja senzor Kinect. Orodje je zelo priročno za hitrejši razvoj in ustvarjanje ponovljivih scenarijev za testiranje in analizo.




Slika 4: Aplikacija Kinect Studio.

Na sliki 9 je prikazan Kinect Studio, ki ima štiri okna:

- glavno okno s časovnim okvirjem in z opcijami za snemanje,
- okno za prikaz barvne slike prikazuje sliko, ki jo pošilja kamera RGB,
- okno za prikaz globinske slike,
- okno za prikaz 3D slike (združena barvna slika in globinska slika).

# 4 Aplikacija geste

Aplikacija geste je prva izmed dveh aplikacij, ki smo ju razvili v okviru diplomske naloge. Za zaznavanje gest smo uporabili senzor Kinect, s katerim sledimo uporabniku s pomočjo skeletnega toka. Napisali smo zunanjo knjižnico »KinectGestureLib«, ki nam pomaga pri realizaciji in detekciji gest. Knjižnica ni del aplikacije in se lahko uporabi tudi v drugih aplikacijah.

## 4.1 Kaj je gesta?

Gesta ali kretnja je ena od oblik neverbalne komunikacije, kjer z gibi telesa, s telesno držo, z obrazno mimiko, glasom, dotikom in očmi sporočamo neko sporočilo. So del našega načina komuniciranja z drugimi ljudmi in jih uporabljamo vzporedno z izrečenimi besedami. Geste so ena najstarejših oblik sporočanja in so vezane na kulturo, veliko vlogo pa imajo predvsem v religiji in duhovnih obredih. Kot smo že omenili, je gesta ena od oblik neverbalne komunikacije, ki je ne smemo enačiti z znakovnim jezikom. Geste so dopolnilo govora, preko katerih posredujemo občutke in misli ali celotna sporočila. Znakovni jezik je za razliko od njih nadomestilo govora, ki ga uporabljajo gluhonemi.

Gesta ima tri faze, to so priprava, poteza in umik. Sporočilo geste se nahaja v potezi, medtem ko priprava in umik označujeta začetek in konec geste. Geste ločimo v več zvrsti. Izpostavili bi dve vrsti gest, ki sta zanimivi z uporabniškega vidika z računalnikom. Prva je simbolna gesta. Simbolne geste se uporabljajo za komunikacijo razpoloženja in ustrezajo izrečenim besedam. Primeri simbolne geste so navzgor obrnjen palec, ki simbolizira dobro opravljeno delo, iztegnjena pokončna dlan simbolizira stop in dlan s tremi iztegnjenimi prsti število tri. Druga je fizična gesta, ki zajema obrazno mimiko, dotik, gibe telesa, glas in oči. Primeri fizične geste so zamah roke za pozdrav, priklon igralca na gledališkem odru, s katerim izrazi hvaležnost, z nasmejanim obrazom pa izrazimo veselje.

## 4.2 Kako zaznamo gesto v aplikaciji

Osredotočili smo se na enoročne geste, in sicer na geste, narejene z desno roko. Zaznavanje gest smo v aplikaciji razdelili na tri dele oziroma faze, kar smo že omenili v zgornjem poglavju. Z mirovanjem roke na enem mestu najavimo prvo fazo »priprava« oziroma začetek geste. Aplikacija mirovanje roke označi z rumenim krožcem. Druga faza je izvajanje geste oziroma »poteza«, kjer uporabnik izvaja gesto. V aplikaciji izvajanje geste označimo z barvnimi krožci. Zadnja faza je prenehanje izvajanja geste ali »umik«, ki ga najavimo z mirovanjem roke.

Gesto v aplikaciji opišemo s točkami, ki jih dodajamo na listo v točno določenem časovnem intervalu (predpostavljena vrednost je 80 milisekund). Vsaka točka vsebuje informacijo o poziciji in smernem premiku. Smer premika točke določimo glede na pozicijo predhodne točke. Imamo pet možnih premikov:

- premik gor - roka se pomika navzgor (v aplikaciji je točka obarvana z modro barvo),
- premik dol - roka se pomika navzdol (v aplikaciji je točka obarvana z zeleno barvo),
- premik levo - roka se pomika levo (v aplikaciji je točka obarvana s sivo barvo),
- premik desno - roka se pomika desno (v aplikaciji je točka obarvana s črno barvo),
- roka je na miru - roka se ne premika v nobeno smer (v aplikaciji je točka obarvana z rumeno barvo).

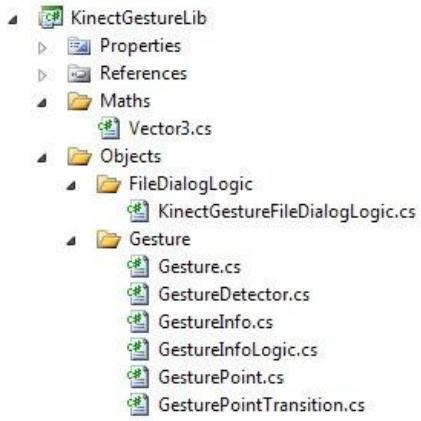
Ko je gesta končana, preverimo prehode smernih premikov točk. Smerne premike točk, pri katerih se je smer spremenila od predhodne točke, dodamo na listo prehodov. Ignoriramo točke, ki imajo enak smerni premik od predhodne točke. Geste zaznamo s primerjanjem list prehodov. Primerjava mora izpolnjevati dva pogoja. Prvi pogoj je enako število prehodov na obeh listah. Drugi pogoj pa je primerjanje prehodov med seboj - primerjamo enakost posameznega prehoda na obeh listah po vrstnem redu. V primeru, da sta oba pogoja izpolnjena, smo zaznali gesto.

## 4.3 Knjižnica KinectGestureLib

Napisali smo knjižnico KinectGestureLib, ki jo vidimo na sliki 10. Uporabili smo jo v aplikaciji geste za realizacijo in detekcijo gest. Knjižnico KinectGestureLib sestavljajo naslednji razredi:

- **Vector3** - ima tri parametre »X«, »Y« in »Z«, ki predstavljajo točko v prostoru.
- **GesturePoint** - hrani pozicijo točke in informacijo o smernem premiku.
- **Gesture** - dodaja točke na listo za opis geste. Točki nastavi pozicijo in smer premika ter življenjsko dobo. Razredu lahko nastavimo parametre, kot so frekvenca dodajanja, življenjska doba točka (dinamično odstranjevanje starih točk) in minimalen premik, ki je potreben, da točka dobi smerni premik.
- **GesturePointTransition** - ima samo en parameter, v katerem hrani podatek o prehodu. Prehod je smerni premik točke.
- **GestureInfoLogic** - ima metodo, s katero generira listo prehodov iz opisnih točk geste v razredu Gesture.
- **GestureInfo** - hrani podatke o gesti, kot so ime, opis in lista prehodov, ki definirajo gesto.

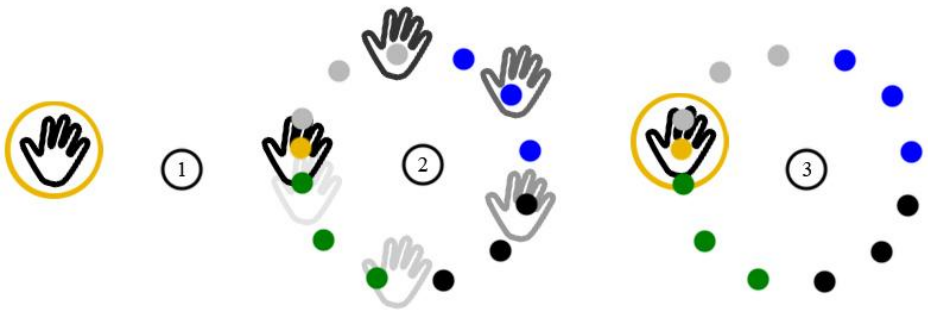
- **KinectGestureFileDialogLogic** - shranjuje in nalaga shranjene geste. Geste se shranijo v binarno datoteko s končnico .gst.




Slika 5: Knjižnica KinectGestureLib.

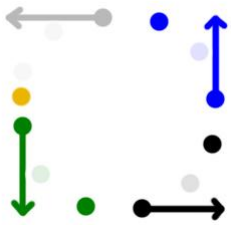
## 4.4 Primer zaznavanja geste krog

V tem poglavju smo opisali vizualni postopek detekcije geste krog. Začetek slednje najavimo z mirovanjem roke na mestu, kjer želimo začeti z njenim izvajanjem. Aplikacija začetek najavi z rumenim krogom, ki ga vidimo na sliki 11 v točki 1. Gesto izvajamo v nasprotni smeri urinega kazalca. Izvajanje geste riše barvne točke, ki predstavljajo smerni premik roke - točka 2. Gesto zaključimo na enaki poziciji, kot smo gesto začeli. Zaključek geste nakažemo z mirovanjem roke - točka 3.




Slika 6: Zaznavanje geste krog - tri faze.

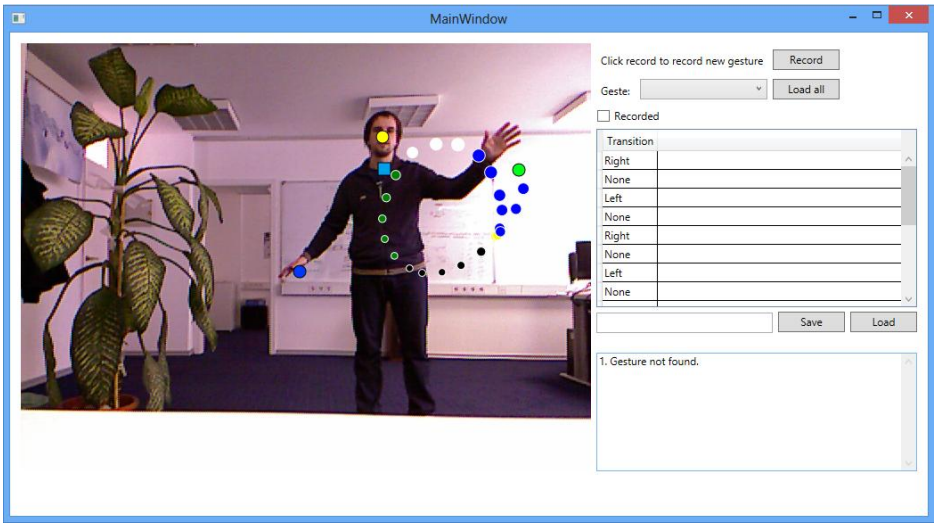
Gesta se je končala. Izvede se klic metode, ki preveri gesto med gestami, ki so bile pred tem naložene v aplikacijo. Metoda pred preverjanjem naredi listo prehodov. Gesta krog ima štiri prehode, ki jih prikazuje slika 12. Prvi prehod je navzdol (zelena puščica), drugi prehod desno (črna puščica), tretji prehod gor (modra puščica) in zadnji četrti prehod levo (siva puščica).



Slika 7: Zaznavanje geste krog – prehodi.

## 4.5 Opis aplikacije geste

Aplikacija je sestavljena iz dveh delov, kot je razvidno na sliki 13. Za prikaz slikovnega toka kamere RGB na levi strani aplikacije smo uporabili knjižnico KinectWpfViewers iz paketa Toolkit.




Slika 8: Geste aplikacija.

Knjižnica vsebuje kontrole, ki nam omogočajo prikaz tokov, ki jih pošilja senzor Kinect. Za prikaz slikovnega toka kamere RGB smo uporabili kontrolo KinectSensorChooser (Programska koda 3) in KinectColorViewer (Programska koda 4). Kontrola KinectSensorChooser poskrbi za povezavo s senzorjem Kinect, kontrola KinectColorViewer pa za prikaz slikovnega toka, ki ga pošilja kamera RGB.

```
<my:KinectSensorChooser Name="kinectSensorChooser1" />
```

Programska koda 3: Kontrola KinectSensorChooser.

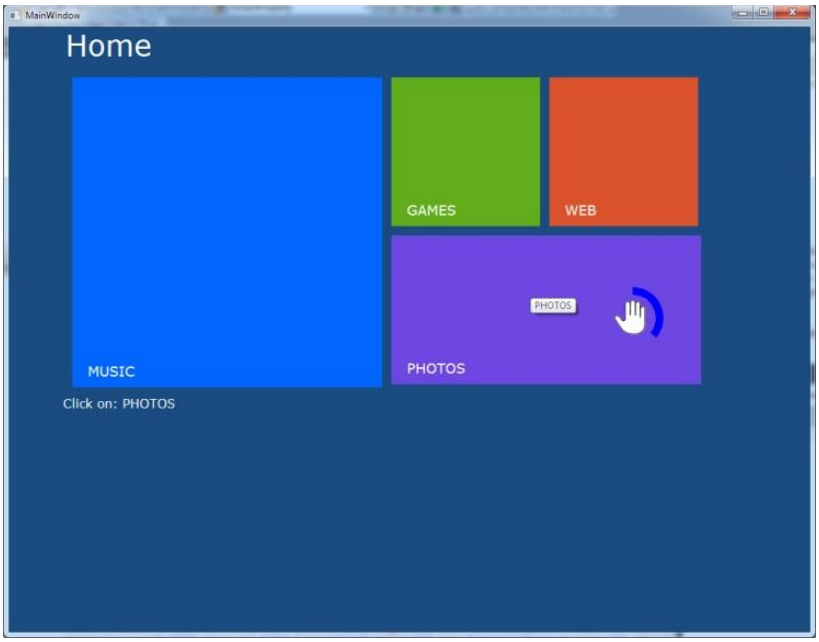
```
<my:KinectColorViewer Name="kinectColorViewer1" Kinect="{Binding
    ElementName=kinectSensorChooser1, Path=Kinect}" DataContext="{Binding}" />
```

Programska koda 4: Kontrola KinectColorViewer.

Desna stran aplikacije predstavlja možnost shranjevanja in nalaganja gest. Geste shranjujemo v binarno datoteko s končnico .gst. Dodali smo opcijo nalaganja posamezne geste ali geste iz datoteke. Naložene geste lahko izberemo iz spustnega menija in po izboru izpišemo v mrežno polje njene prehode. Na koncu je polje, kjer se izpisujejo imena gest, ki jih je aplikacija zaznala.

# 5 Brezdotični brskalnik

V drugem delu diplomske naloge smo razvili aplikacijo za prikazovanje spletnih vsebin, kjer smo računalniško miško nadomestili s senzorjem Kinect. Aplikacijo brezdotični brskalnik prikazuje slika 14. Namen aplikacije je prikaz spletne strani in nadomeščanje računalniške miške s senzorjem Kinect kot vhodne naprave za klikanje. Za prikaz spletnih strani smo se odločili zaradi njene široke uporabe in preproste interakcije. Na spletnih straneh lahko prikažemo široko paleto različnih tipov vsebin, kot so spletne galerije, spletni katalogi, video predstavitve, animacije, tekstovne vsebine itd. Interakcija s temi vsebinami je v večini primerov, če se osredotočimo na osebni računalnik, računalniška miška.




Slika 9: Aplikacija brezdotični brskalnik.

Aplikacija brezdotični brskalnik je sestavljena iz glavnega okna, kjer s kontrolo WebBrowser prikažemo vsebino spletne strani in kontrolo, s katero nadomestimo kazalec računalniške miške (ikona desne roke). Osredotočili smo se na simuliranje ene miške z desno roko.

Za potrebe testiranja smo napisali testno spletno stran, ki jo prikazuje slika 14. Spletna stran vsebuje poimenovna polja, po katerih lahko klikamo. S senzorjem Kinect simuliramo klik in premik miškinega kazalca po elementih spletne strani. Posamezno polje na testni spletni strani ima tri pomembne parametre, to so »onclick«, »onmouseover« in »onmouseout«. Primer kode in parametrov vidimo pri Programski kodi 5.

```
<a id="photos" class="button butW2H1 bPurple" title="PHOTOS"
        onclick="ExecuteClick(this);"
        onmouseover="javascript:window.external.InvokeMouseOver();"
        onmouseout="javascript:window.external.InvokeMouseOut();" >
              <span>PHOTOS</span>
 </a>
```

Programska koda 5: Polje PHOTOS na testni spletni strani.

Akcija »onmouseover« se zgodi takrat, ko uporabnik premakne roko na eno izmed polj in zadrži pozicijo roke. Akcija izvede klic metode v posebnem razredu, ki smo ga označili kot razred COM [3]. Ta označitev nam omogoča, da lahko na spletni strani kličemo metode znotraj aplikacije. Na spletni strani pokličemo metodo v aplikaciji InvokeMouseOver (Programska koda 6) s klicem »javascript:window.external.InvokeMouseOver();"« (Programska koda 5). Klic metode zažene izvajanje odštevalnika, ki odšteva čas do klika. Odštevalnik, ki ga vidimo na sliki 14, je del kontrole, s katero smo nadomestili kazalec miške, je modre barve v obliki kroga in odšteva v smeri urinega kazalca.

```
public void InvokeMouseOver()
{
       OnJavascriptInvoked(this, new MouseEventArgs(MouseEnum.MouseOver));
}

public void InvokeMouseOut()
{
       OnJavascriptInvoked(this, new MouseEventArgs(MouseEnum.MouseOut));
}
```

Programska koda 6: Metodi InvokeMouseOver() in InvokeMouseOut().

Ko odštevalnik konča z odštevanjem, se izvede klic dogodka, kjer simuliramo klik miške s klicem funkcije SendInput [11]. Funkcija SendInput je posebna funkcija, ki pošilja ukaze vhodnim napravam, kot sta tipkovnica in miška, aplikacijam, ki tečejo v operacijskem sistemu Windows. Klik miške povzroči akcijo »onclick«, ki izvede klic funkcije JavaScript [8] »ExecuteClick(this)« (Programska koda 7). Funkcija izpiše na strani ime polja, na katerem smo simulirali klik.

```
<script type="text/javascript">
 function ExecuteClick(obj) {
   document.getElementById('info1').innerHTML = 'Click on: ' + obj.title + ' ';
 }
</script>
```

Programska koda 7: Funkcija JavaScript ExecuteClick(this).

Klik na polju lahko prekličemo na dva načina. Prvi način je premik izven polja, s čimer povzročimo akcijo »onmouseout«, ki pokliče metodo »InvokeMouseOut()« v aplikaciji za ustavitev odštevalnika. Drugi način je premik roke z mesta, kjer se je zagnal odštevalnik.

# 6 Sklep

V okviru diplomske naloge smo izdelali dve aplikaciji, ki prikazujeta možnost uporabe senzorja Kinect kot brezdotične vhodne naprave. Prva aplikacija geste omogoča prepoznavanje gest. Zaradi lažjega testiranje smo se osredotočili le na enoročne geste, narejene z desno roko. Druga aplikacija deluje kot spletni brskalnik, ki prikazuje prilagojene spletne strani. Predstavili smo rešitev, kako lahko s senzorjem Kinect nadomestimo oziroma simuliramo računalniško miško. Pri obeh aplikacijah je seveda še veliko prostora za izboljšave.

## 6.1 Predlogi za nadaljnje delo

Aplikacijo geste lahko nadgradimo, da bo zaznala tudi geste leve roke. Nadgradnja za prepoznavanja gest leve roke bi bila preprosta. Podvojili bi logiko, ki prepoznava geste desne roke, in popravili vhodne parametre na parametre leve roke, ki jih dobimo iz skeletnega toka.

Za detekcijo dvoročnih gest (leve in desne roke hkrati) pa bi potrebovali nekaj prilagoditev. Predpostavimo, da je dvoročna gesta izvedena hkrati z levo in desno roko. Preprosta rešitev bi bila primerjanje časovne razlike med začetkom obeh gest. V primeru, da sta se obe gesti začeli ob enakem času, bi gesto smatrali kot dvoročno gesto. Dopolniti bi bilo potrebno tudi metodo za prepoznavanje gest. Trenutna metoda prepoznava samo eno gesto naenkrat, zato bi jo morali prilagoditi oziroma dopolniti tako, da bi lahko prepoznavala dve ali več gest hkrati.

Kot smo opisali, našo rešitev za prepoznavanja gest, začetek in konec, najavimo z mirovanjem roke na enem mestu. Tak način prepoznavanja gest je zelo tog, želeli bi boljše, dinamično prepoznavanje, kjer sistem iz niza gibov našega telesa razbere začetek, gesto in konec. Ena od rešitev bi bila, da bi dodali življenjsko dobo točkam, ki predstavljajo gesto. Na ta način bi odstranili stare točke in iskali gesto samo v aktualnih točkah. Ker v večini primerov uporabniki podajajo geste z iztegnjenimi rokami, bi dodali v točke še oddaljenost posamezne točke od našega telesa. Ta podatek bi nam pomagal določiti začetek in konec geste. Delno je to v aplikaciji že implementirano.

Poleg zgoraj omenjenih predlogov bi dodali tudi boljše shranjevanju gest. Geste v aplikaciji shranjujemo v binarni zapis. Shranimo cel objekt, ki predstavlja informacijo o gestah. Ta zapis je težko berljiv in prenosljiv. Za branje tovrstnih zapisov moramo imeti metode, ki znajo prebrati zapis nazaj v enak objekt kot pred zapisom. Če se med razvojem objekt spremeni, postanejo shranjene geste neuporabne. Zato bi nadgradili shranjevanje gest v XML ali Jason datoteko. V primeru, da pride do sprememb pri pisanju in branju gest v aplikaciji, lahko posamezno gesto popravimo v tekstovnem urejevalniku.

Pri aplikaciji brezdotični brskalnik bi lahko olajšali razvoj oziroma omogočili interakcijo s spletnimi stranmi, ki niso prilagojene za brezdotični brskalnik. Kot smo opisali, za simuliranje klika potrebujemo posebne klice metod s spletne strani v aplikacijo. To bi lahko rešili tako, da bi pred prikazom spletne strani vsem potrebnim elementom nastavili potrebne parametre, ki jih aplikacija brezdotični brskalnik potrebuje za interakcijo.

Brezdotični brskalnik ima samo eno gesto klik. Poleg te geste bi lahko dodali še druge. Z gestami, kot so naprej –zamah levo, nazaj –zamah desno in gesta domov – z roko orišemo krog. Za implementacijo teh gest bi uporabili knjižnico KinecGestureLib, ki smo jo napisali za prepoznavanje gest pri prvi aplikaciji. Implementacija bi bila preprosta, geste pa bi posneli v aplikaciji geste in jih pred zagonom naložili v brezdotični brskalnik.

Namesto prilagojenega brskalnika, ki omogoča uporabo s senzorjem Kinect, bi napisali aplikacijo, ki bi s pomočjo spletne tehnologije WebSocket [13] pošiljala ukaze spletnim brskalnikom, kot so Chrome, Mozilla Firefox ali Internet Explorer. Tako aplikacijo smo razvili s sodelavci v podjetju Neolab. Razvili smo interaktivni oglasni kiosk, ki prepleta geste iz senzorja Kinect in spletno galerijo v tehnologiji HTML5.

# Kazalo slik

Slika 1: Senzor Kinect. ..... 3
Slika 3: Infrardeča projekcija projektorja. ..... 4
Slika 4: Senzorji gibanja Xtion, Xtion PRO in Ktion PRO LIVE. ..... 6
Slika 5: Microsoft Visual C# 2010 Express. ..... 7
Slika 1: Globinska slika. ..... 9
Slika 2: Točke sklepov. ..... 10
Slika 3: Aplikacija Toolkit Browser. ..... 12
Slika 4: Aplikacija Kinect Studio. ..... 12
Slika 5: Knjižnica KinectGestureLib. ..... 17
Slika 6: Zaznavanje geste krog - tri faze. ..... 17
Slika 7: Zaznavanje geste krog – prehodi. ..... 18
Slika 8: Geste aplikacija. ..... 18
Slika 9: Aplikacija brezdotični brskalnik. ..... 21

# Kazalo izvorne kode

Programska koda 1: Poišče prvi senzor, ki je povezan na računalnik. ..... 8
Programska koda 2: Objekt TransformSmoothParameters s parametri za glajenje. ..... 11
Programska koda 3: Kontrola KinectSensorChooser. ..... 18
Programska koda 4: Kontrola KinectColorViewer. ..... 19
Programska koda 5: Polje PHOTOS na testni spletni strani. ..... 22
Programska koda 6: Metodi InvokeMouseOver() in InvokeMouseOut() ..... 22
Programska koda 7: Funkcija JavaScript ExecuteClick(this). ..... 23

# Literatura

[1] (2012) .Net Framework. Dostopno na:
http://en.wikipedia.org/wiki/.NET_Framework

[2] (2012) Common Language Runtime. Dostopno na:
http://en.wikipedia.org/wiki/Common_Language_Runtime

[3] (2012) Component Object Model. Dostopno na:
http://en.wikipedia.org/wiki/Component_Object_Model

[4] (2012) Developer downloads. Dostopno na:
http://www.microsoft.com/en-us/kinectforwindows/develop/developer-downloads.aspx

[5] (2012) How It Works: Xbox Kinect. Dostopno na:
http://www.jameco.com/Jameco/workshop/howitworks/xboxkinect.html

[6] (2012) How Microsoft Kinect Works - Kinect Software Learns from "Experience". Dostopno na:
http://electronics.howstuffworks.com/microsoft-kinect3.htm

[7] (2012) Java Language. Dostopno na:
http://en.wikipedia.org/wiki/Java_(programming_language)

[8] (2012) Javascript. Dostopno na:
http://en.wikipedia.org/wiki/JavaScript

[9] (2012) Alumnus Dr Jamie Shotton and the development of Kinect for Xbox 360. Dostopno na:
http://www.eng.cam.ac.uk/news/stories/2011/Xbox_Kinect/

[10] (2012) Kinect. Dostopno na:
http://en.wikipedia.org/wiki/Kinect

[11] (2012) SendInput function. Dostopno na:
http://msdn.microsoft.com/en-us/library/windows/desktop/ms646310(v=vs.85).aspx

[12] (2012) Visual C# Language. Dostopno na:
http://msdn.microsoft.com/en-us/library/aa287558(v=vs.71).aspx

[13] (2012) WebSocket. Dostopno na:
http://en.wikipedia.org/wiki/WebSocket

[14] (2012) Working with Kinect Studio. Dostopno na:
http://msdn.microsoft.com/en-us/magazine/jj650892.aspx

[15] (2012) Xtion - Motion Sensor for PC. Dostopno na:
http://www.asus.com/Multimedia/Motion_Sensor/Xtion

[16] (2012) Xtion PRO - Use Xtion PRO developer solution to make motion-sensing applications and games. Dostopno na:
http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO

[17] (2012) Xtion PRO LIVE - The world's first PC exclusive motion sensing development solution. Dostopno na:
http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO_LIVE