VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

OVLÁDÁNÍ DESKTOPU POMOCÍ SENZORU KINECT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE BC. ZSOLT CZOMPÁL
AUTHOR

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

OVLÁDÁNÍ DESKTOPU POMOCÍ SENZORU KINECT

KINECT-BASED DESKTOP CONTROLLER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE BC. ZSOLT CZOMPÁL
AUTHOR

VEDOUCÍ PRÁCE ING. PAVEL ŽÁK
SUPERVISOR

BRNO 2012

Abstrakt

Cílem této diplomové práce bylo vytvořit uživatelské rozhraní, které umožní ovládat počítač pomocí hlasových povelů, statických póz a dynamických gest. Obsahuje implementaci přirozeného uživatelského rozhraní, které emuluje funkci klávesnice a myši. Na rozpoznávání a sledování uživatele v prostoru je v této práci použit senzor Kinect. Pro implementaci jsem zvolil platformu Windows, použitý vývojový nástroj je Kinect for Windows SDK s použitím programovacího jazyka C#. Při implementaci a testování rozhraní byl použit senzor Kinect for Xbox 360. Výsledkem práce je přirozené uživatelské rozhraní, které umožní ovládat desktopové aplikace bez použití klávesnice a myši, a je pro uživatele pohodlné a přívětivé.

Abstract

The aim of this master's thesis was to create a user interface that allows controlling the PC using voice commands, static poses and dynamic gestures. It contains an implementation of a natural user interface that emulates the functionality of the keyboard and the mouse. In this thesis for detecting and tracking the user was used the sensor Kinect. For implementation I have chosen platform Windows and the Kinect for Windows SDK with programming language C#. During the implementation and testing of the interface, sensor Kinect for Xbox 360 was used. The result of this master's thesis is a natural user interface that allows controlling desktop applications without using keyboard or mouse, and that is comfortable and user-friendly.

Klíčová slova

KinectDesktop, přirozené uživatelské rozhraní, NUI, póza, gesto, Kinect, Kinect for Windows SDK

Keywords

KinectDesktop, natural user interface, NUI, pose, gesture, Kinect, Kinect for Windows SDK

Citace

Czompál Zsolt: Ovládání desktopu pomocí senzoru Kinect, diplomová práce, Brno, FIT VUT v Brně, 2012

Ovládání desktopu pomocí senzoru Kinect

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Pavla Žáka.
Další informace mi poskytl Ing. Pavel Jetenský.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Zsolt Czompál
23. 5. 2012

Poděkování

Poděkování v první řade patří mému vedoucímu práce Ing. Žákovi za jeho odborné rady, trpělivost a podporu, za projevený zájem a lidský přístup. Dále bych chtěl poděkovat Ing. Jetenskému za poskytnutí jeho odborných publikací. Poděkovat bych chtěl kolegům z fakulty, kteří mi poradili při řešení problematičtějších kapitol mé práce. V neposlední řade bych chtěl poděkovat Ing. Katalin Névery za gramatickou korekturu a nemalou psychickou podporu.

© Zsolt Czompál, 2012
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Obsah 1
1 Úvod 3
2 HCI – Interakcia človek - počítač 4
2.1 Dnešné grafické užívateľské rozhrania 6
2.2 NUI – Prirodzené užívateľské rozhranie 8
2.2.1 Hlasové NUI rozhranie 9
2.2.2 Kinetické NUI rozhranie 10
2.3 Rozpoznávanie pohybov človeka založené na počítačovom videní 12
2.3.1 Základné koncepty rozpoznávania pohybu v obraze 12
2.3.2 Rozpoznanie statických póz 13
2.3.3 Rozpoznávanie dynamických gest 14
3 Senzor Kinect 18
3.1 Komponenty senzoru Kinect 18
3.2 Structured Light kamerové systémy 20
3.3 Ovládače a vývojové nástroje ku Kinectu 22
3.3.1 OpenKinect a libfreenect 23
3.3.2 CL NUI Platform 23
3.3.3 OpenNI a OpenNI framework 23
3.4 Kinect for Windows SDK 25
3.4.1 Vývoj aplikácií v Kinect for Windows SDK 26
4 Prirodzené užívateľské rozhranie a Kinect 29
4.1 Spôsoby ovládania klasických desktopových aplikácií pomocou Kinectu 29
4.1.1 Ovládanie klasických desktopových aplikácií 29
4.1.2 Grafické užívateľské rozhranie desktopových aplikácií 30
4.1.3 Ovládanie desktopových aplikácií pohybmi užívateľa 31
4.1.4 Ovládanie desktopových aplikácií ľudskou rečou 33
4.2 Návrh prirodzeného užívateľského rozhrania pracujúceho s Kinectom 35
4.2.1 Ovládanie kurzoru myši 35
4.2.2 Ovládanie desktopových funkcií 36
4.2.3 Ovládanie textového kurzoru 38
4.2.4 Prepínanie režimov rozhrania 39
5 Prirodzené užívateľské rozhranie KinectDesktop 40
5.1 Hlavné okno rozhrania KinectDesktop 40
5.2 DesktopController – ovládanie kurzoru myši a desktopových funkcií 42

5.2.1 PoseDetector – detekcia statických póz 42
5.2.2 SwipeDetector – detekcia gesta swipe 43
5.2.3 Ovládanie kurzoru myši 44
5.2.4 Ovládanie funkcií desktopu a aplikácií 45
5.2.5 Rozpoznávanie gesta na zmenu režimu ovládania 45
5.3 KeyboardController – ovládanie virtuálnej klávesnice 46
5.3.1 Virtuálna klávesnica a virtuálne klávesy 46
5.3.2 Grafické rozhranie virtuálnej klávesnice 47
5.3.3 Databáza virtuálnych znakov 48
5.3.4 Emulácia funkcií klávesnice 49
6 Výsledky a vyhodnotenie práce 52
7 Záver 54

1 Úvod

Skúmanie komunikácie medzi človekom a počítačom je významnou oblasťou vedy informačných technológií. Táto technológia sa v posledných dvadsiatich rokoch vyvíja nesmierne rýchlo, a čoraz viac ľudí používa počítače denne tak pri riešení problémov pri práci, ako aj v osobnom živote. Rozšírenie počítačov v takejto miere so sebou automaticky prinieslo aj potrebu zjednodušenia ich ovládania. Jedným z najdôležitejších vynálezov, ktoré toto zjednodušenie vo veľkom podporujú, je podľa môjho názoru dotyková obrazovka. Prvé počítače s dotykovou obrazovkou sa objavili už v osemdesiatich rokoch minulého storočia, ich masové rozšírenie sa však začalo až v novom tisícročí. Dnes už každý môže vlastniť počítač s dotykovou obrazovkou, a to vo forme chytrého telefónu. Ovládanie týchto telefónov je veľmi jednoduché, pracuje sa s nimi oveľa jednoduchšie, než s klasickými telefónmi, ktoré majú tlačidlá. Táto skutočnosť obrátila pozornosť vývojárov na skúmanie možností podobného zjednodušenia komunikácie medzi užívateľom a počítačom.

Vývoj sa začal hneď vo viacerých smeroch. Samozrejme sa pokračovalo vo zdokonaľovaní dotykových obrazoviek, ale vznikli aj projekty, ktoré využívali rozpoznávania ľudskej reči či pohybov. Spoločným cieľom týchto projektov je dosiahnutie priamej interakcie človeka a počítača, bez použitia periférií ako sú klávesnica a myš.

V mojej diplomovej práci sa detailnejšie zaoberám rozpoznávaním ľudskej reči a pohybov, a ovládaním počítača ich pomocou. Predmetom tejto práce sú prirodzené užívateľské rozhrania a senzor Kinect. Jej hlavným cieľom je vytvoriť rozhranie, ktoré pomocou senzoru Kinect emuluje funkciu klávesnice a myši, a tak umožní ovládať rôzne desktopové aplikácie pomocou hlasu a pohybu.

Moja práca sa delí na dve hlavné časti: na časť teoretickú a na časť praktickú. V prvej časti sa zaoberám teóriou prirodzeného užívateľského rozhrania a senzorom Kinect, v druhej časti sa venujem návrhu a implementácie takého rozhrania.

Úvodná druhá kapitola mojej práce má názov HCI – Interakcia – človek počítač. Zaoberám sa v nej teoretickými základmi prirodzených užívateľských rozhraní. Detailnejšie popíšem rôzne možné pohyby užívateľa, spôsoby ich detekcie a možnosti ovládania počítača pomocou týchto pohybov.

Tretia kapitola je venovaná senzoru Kinect. Predstavím jeho zabudované komponenty, základnú myšlienku a technológiu jeho funkcie. Ďalej sa tu zaoberám ovládačmi a vývojovými nástrojmi spomínaného senzoru, detailnejšie z nich popíšem vývojový nástroj Kinect for Windows SDK, ktorý som používal pri implementácii môjho rozhrania v praktickej časti tejto práce.

Ďalšia kapitola má názov Prirodzené užívateľské rozhranie a Kinect. V prvej časti tejto kapitoly sa venujem možnostiam ovládania desktopových aplikácií pomocou senzoru Kinect. Podrobne popíšem spôsoby ovládania pomocou statických póz, dynamických gest a hlasových povelov. V druhej časti kapitoly sa venujem integrácii uvedených spôsobov ovládania do prirodzeného užívateľského rozhrania, a uvádzam návrh rozhrania implementovaného v praktickej časti mojej práce.

V nasledujúcej kapitole predstavím riešenia, ktoré som používal pri implementácii môjho rozhrania. Predstavím grafickú podobu rozhrania a obrázkami ilustrujem jej funkciu a použitie.

Posledná kapitola je venovaná vyhodnoteniu výsledkov mojej práce, ohodnotím v nej výslednú efektivitu a funkčnosť navrhovaného a implementovaného rozhrania a jeho prívetivosť z hľadiska užívateľa.

Z dôvodu, že som svoj semestrálny projekt a svoju diplomovú prácu nespracoval v jednom akademickom roku, moja diplomová práca nenadväzuje na môj semestrálny projekt.

2 HCI – Interakcia človek - počítač

Veda o interakcii človek - počítač – *Human Computer Interaction* (ďalej len *HCI*) – je veľmi rozsiahla, združuje rôzne oblasti technických a humánnych oborov. Aby sme mohli navrhnúť a vytvoriť kvalitné užívateľské rozhranie (*User Interface - UI*) pre prácu s výpočtovou technikou, musíme dvojcestnú komunikáciu medzi človekom a počítačom skúmať z viacerých pohľadov. Počítačová veda skúma proces vývoja aplikácií a návrh rozhrania; sociológia a antropológia sa zaoberá vzájomným pôsobením medzi technológiou, prácou a organizáciou; psychológia je zameraná na analýzu správania užívateľov v priebehu interakcie s počítačom a na aplikáciu teoretických kognitívnych procesov; ergonómia hľadá interaktívne a pre užívateľa prívetivé, pohodlné a efektívne rozhranie (1).

Presne definovať HCI je veľmi náročné kvôli rôznorodosti tejto oblasti vedy. Je nutné pokryť všetky aspekty interakcie medzi človekom a počítačom o ktorých som sa zmienil v predchádzajúcej kapitole. Najjednoduchšia a najvšeobecnejšia definícia, ktorá pokryje všetky oblasti HCI by mohla znieť nasledovne: *HCI je veda, ktorá skúma komunikáciu medzi človekom a počítačom*. Konkrétnejšiu definíciu nájdeme v (1): „Interakcia človek – počítač je disciplína zaoberajúca sa návrhom, hodnotením a implementáciou interaktívnych výpočtových systémov pre užívateľov, a študuje významnejšie javy vyskytujúce sa počas interakcie.“ Táto práca sa zaoberá hlavne predmetom prvej časti definície, t.j. návrhom, implementáciou a hodnotením interakčného systému medzi človekom a počítačom. Nie je ale možné obísť ani vynechať ostatné oblasti interakcie. Myslím tým predovšetkým to, že pri návrhu je veľmi dôležitá prívetivosť a efektívnosť navrhovaného systému. Jednoduchý a ľahko pochopiteľný interakčný systém môže kladne ovplyvniť úspech výslednej aplikácie, kým nerozmyslený a neergonomický systém môže spôsobiť aj veľké škody[1].

V rannej ére počítačov skúmanie interakcie človek – počítač ešte nehralo v počítačových vedách významnú rolu. Keď odhliadneme od niekoľkých priekopníkov – napr. Vannevar Bush v roku 1945 vydal esej *As We May Think*[2] – výskum v oboru HCI sa začal len v šesťdesiatich rokoch 20-teho storočia. Prvé semináre s touto tematikou nasledovali zborníky a časopisy popisujúce rôzne aspekty komunikácie medzi človekom a počítačom (2). Nové prvky užívateľských rozhraní, ktoré sa objavili v šesťdesiatych a v sedemdesiatych rokoch priniesli nové výzvy v oblasti HCI. Grafické prvky a primitívne grafické užívateľské rozhrania objasnili, že výskum v oblasti HCI sa môže stať veľmi dôležitým pri návrhu komerčných aplikácií.

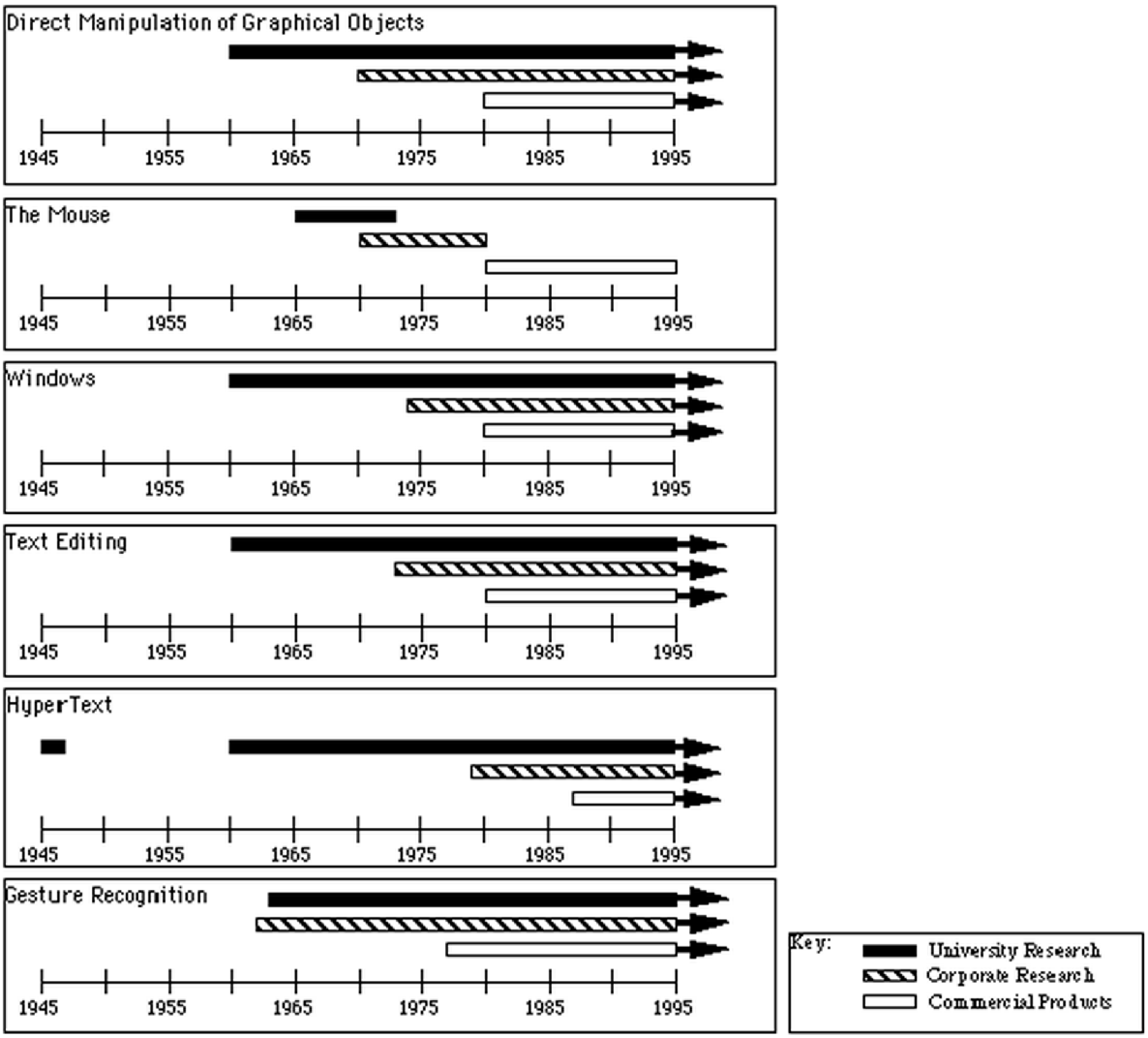
Interakcia človek – počítač znamená z pohľadu počítačových vied a neskôr aj z pohľadu počítačovej grafiky návrh a implementáciu užívateľských rozhraní. Prvé počítače také rozhranie ešte nemali, ako vstupné a výstupné rozhranie používali *dierne štítky*. Užívateľ zadal vstupný štítok do počítača, a po výpočtu obdržal výsledok. Interakcia s počítačom v reálnom čase, alebo zmena preddefinovanej úlohy v priebehu výpočtu bola nemožná. „Programovanie“ diernych štítkov bolo veľmi náročné, chyby na štítku boli neprípustné, akýkoľvek malý rozdiel vo vstupu by totiž znamenal celkom odlišný alebo chybný výsledok. Keď si uvedomujeme, že na týchto strojoch aj jednoduchšie výpočty trvali viac hodín, tak môžeme skonštatovať, že rozhranie s diernymi štítkami nebolo pre užívateľa vôbec prívetivé.

Veľkú zmenu v komunikácii medzi človekom a počítačom znamenalo rozšírenie rozhrania príkazovým riadkom (*Command Line Interface - CLI*), ktorý postupne vystriedal dierne štítky

[1] Three Mile Island accident (http://en.wikipedia.org/wiki/Three_Mile_Island_accident).
[2] Memex (http://en.wikipedia.org/wiki/Memex).

a výstupné zariadenie zvané *batch monitor*. Príkazový riadok je textové rozhranie typu príkaz – odpoveď s vlastnou syntaxou a sémantikou jednotlivých príkazov. Toto nové rozhranie urýchlilo interakciu s počítačom, užívateľ nemusel čakať hodiny alebo dni na odpoveď systému, komunikácia sa viac-menej odohrávala v reálnom čase. Napriek pozitívnym prínosom príkazového riadku potreboval užívateľ veľké množstvo príkazov na obsluhu počítača. Napriek tomu, že rozhranie sa začalo rozšíriť ešte s operačnými systémami *Multics* a *Unix*, dodnes je používané v moderných operačných systémoch Unix/Linux/Mac OS (*shell*-y) a Windows (*PowerShell*). (3)

Teoretický vývoj nového rozhrania založeného na grafických prvkoch a vektorovej grafike sa začal už v šesťdesiatych rokoch, hlavne vo výskumných laboratóriách univerzít. Na obrázku 1 je znázornený časový vývoj jednotlivých technológii (história vývoja v (4)).



Obrázok 1 Približné časové údaje reprezentujú vývoj jednotlivých technológii (4)

Prví predkovia dnešných grafických užívateľských rozhraní (*Graphic User Interface – GUI)* sa objavili v sedemdesiatych rokoch minulého storočia. Priekopníkom technológie GUI bola firma *Xerox* s počítačom *Alto*, ktorý mal bitmapovú obrazovku a myš. Užívateľské rozhranie obsahovalo podobné prvky ako dnešné operačné systémy: ikony, okná, tlačítka a scrollbar.

Medzi najúspešnejšie komerčné produkty, ktoré používali prvky GUI predstavené v *Alto*, môžeme zaradiť pracovnú stanicu firmy *Sun Microsystems* s operačným systémom Unix, stroj *Amiga 1000* od firmy *Amiga* a počítač *Macintosh* od *Apple*. Dnešná najväčšia firma na poli operačných

systémov *Microsoft* dokázala presadiť vlastný systém s grafickým rozhraním (*Windows 1.0* bol neúspešný) až na začiatku deväťdesiatych rokov (*Windows 3.0/3.1*). (3)

2.1 Dnešné grafické užívateľské rozhrania

Základy dnešných rozhraní operačných systémov na osobné počítače boli vyvinuté v sedemdesiatych a osemdesiatych rokoch minulého storočia. Novšie systémy len vylepšili stávajúcu podobu grafického rozhrania novými funkciami, hlavná myšlienka o interakcii medzi človekom a počítačom sa nezmenila. Vstupné zariadenia sú bežná klávesnica, myš (na notebookoch touchpad/trackpoint), neobvykle sa stretneme s grafickými tabletmi, kresliacimi perami a so špecializovanými hernými zariadeniami. Vo všetkých prípadoch (s výnimkou touchpadu) je komunikácia s počítačom indirektná: je nutné, aby sme hýbali myšou alebo stlačili klávesu na klávesnici.

Dotykové obrazovky v mobilných zariadeniach priniesli potrebu nového prístupu ku grafickému užívateľskému rozhraniu. U prvých mobilných telefónov a tabletov rozpoznali obrazovky naraz len jeden dotyk, a tak boli možnosti programátorov značne obmedzené. V týchto zariadeniach nájdeme hlavne hybridné rozhrania, ktoré vznikli modifikáciou nedotykových rozhraní. Veľkým prielomom na trhu mobilných telefónov a tabletov bolo vydanie prvého *iPhone*-u od firmy *Apple* s viacdotykovou obrazovkou a s úplne prepracovaným rozhraním. Možnosť detekcie viacerých dotykov naraz umožňovala programátorom integrovať do systému komplexnejšie dotykové gestá. Najpoužívanejšie gestá sú napríklad *pinch/unpinch*[3] alebo priblíženie a rotácia s dvomi prstami. Rozhranie *iPhone*-u sa stalo akýmsi štandardom na trhu mobilných zariadení. Ovládanie telefónov pomocou gest sa stalo jednoduchším, ako pohybovať kurzorom na obrazovke. Čas potrebný k zvyknutia si na nové rozhranie bol zanedbateľný. Môžeme teda skonštatovať, že ovládanie nového rozhrania sa stalo prirodzeným pre užívateľa, a s viacdotykovou obrazovkou je celková ergonómia systému vyššia.

V osemdesiatych a na začiatku deväťdesiatych rokov počítače používali hlavne informatici, ktorí disponovali s príslušnými technickými vedomosťami potrebnými k ovládaniu operačného systému cez integrované užívateľské rozhranie. To znamená, že pred masívnym rozšírením osobných počítačov (na prelomu 20. a 21. storočia) užívatelia neboli úplní laici v počítačových vedách: vedeli „čo sa v počítači odohráva“, keď spustia príkaz z príkazového riadku alebo stlačia tlačidlo na GUI aplikácii. Pri návrhu užívateľských rozhraní mohli programátori predpokladať, že užívateľ má isté technické vedomosti o počítačoch a má skúsenosť s nejakým UI rozhraním. V dnešnej dobe skoro v každej domácnosti nájdeme aspoň jeden počítač. Od domácich užívateľov však nemôžeme vyžadovať, aby používali príkazový riadok, alebo komplikovanejšie grafické užívateľské rozhrania. Pri návrhu nových GUI rozhraní je preto nutné počítať aj s týmito technicky nevzdelanými užívateľmi, len tak môžeme navrhnúť rozhranie, ktoré bude prívetivé a jednoduché pre najväčšiu možnú skupinu užívateľov. Preto budú GUI rozhrania čím ďalej jednoduchšie, budú obsahovať menší počet statických prvkov a viac dynamických objektov. Dobrým príkladom tohto zjednodušovania je nové rozhranie operačného systému *Windows 8* (5) alebo nové prostredie *Unity* v distribúcii Linuxu, *v Ubuntu 11.04*[4].

V predchádzajúcich dvoch odsekoch som sa pokúsil poukázať na to, že jednoduchosť a prirodzenosť pre užívateľa by mali byť hlavnými vlastnosťami moderného užívateľského rozhrania. Kým prirodzenosť je dosiahnuteľná detailným návrhom a postupným testovaním so skutočnými

[3] Wikipedia: Multi-touch. (http://en.wikipedia.org/wiki/Pinching_(technology)#Multi-touch_gestures).
[4] Unity (http://unity.ubuntu.com/about/).

užívateľmi, dosiahnuť to, aby bolo rozhranie prirodzené, môže byť komplikovanejšie. Dôležité je, aby interakcia medzi človekom a počítačom bola direktná, t.j. aby sme pre túto komunikáciu nepotrebovali ďalšie zariadenia (myš, klávesnica, herný ovládač). To je možné dosiahnuť napríklad hlasovými povelmi, hýbaním časťami tela alebo rôznymi gestami rúk.

Zariadenia pre snímanie vyššie popísaných možností ovládania sú dnes už bežne dostupné: mikrofón, kamery, infračervené snímače, pohybové čidlá alebo špecializované senzory *Microsoft Kinect*, *Sony PlayStation Eye*, *Asus Xtion PRO*. Použitím týchto snímačov je možné dosiahnuť, aby interakcia s počítačom bola priama a intuitívna.

2.2 NUI – Prirodzené užívateľské rozhranie

Na konci predchádzajúcej kapitoly som rozobral hlavné vlastnosti najnovších užívateľských rozhraní, ktoré charakterizujú tzv. *NUI* rozhranie (*Natural User Interface* – prirodzené užívateľské rozhranie).

Pod pojmom prirodzeného užívateľského rozhrania si na základe predchádzajúcej kapitoly môžeme predstaviť mnoho, často veľmi odlišných vecí. Formálnejšia definícia pojmu by mala objasniť koncept prirodzenosti u rozhraní takého typu:

„Prirodzené užívateľské rozhranie je rozhranie navrhnuté takým spôsobom, aby využíval existujúce schopnosti a skúsenosti používateľa pre priamu interakciu s obsahom.“ (6)

Tak, ako u rozhraniu typu príkazový riadok a grafické rozhranie, návrh je veľmi dôležitý aj v prípade prirodzeného interakčného systému. Pri návrhu je nutné zachovať základné vlastnosti rozhrania, ktoré priamo vychádzajú z vyššie uvedenej definície: využívanie existujúcich schopností používateľa a priamosť pri interakcii. Okrem dosiahnutia týchto vlastností musí návrh počítať aj požiadavkami užívateľov. NUI rozhranie musí byť jednoduché, a užívateľ si musí vedieť osvojiť jeho používanie v krátkom čase. Z druhej časti definície je zrejmé, prečo sa môže NUI zdať užívateľovi prirodzenejšie ako príkazový riadok alebo grafické UI:

- Užívateľ využíva vlastné schopnosti a skúsenosti pri interakcii s počítačom. Nemusí sa naučiť nové postupy, presné umiestnenie funkcie v ponuke alebo príkazy a ich argumenty. Práca s NUI rozhraním je intuitívna, v mnohých prípadoch je založená na interakciách medzi ľuďmi alebo medzi človekom a fyzickými predmetmi. Dobrým príkladom sú rozhrania dnešných dotykových telefónov: dotykom vyberieme položku z ponuky, priblížime si obrázok dvomi prstami. Prirodzenosť týchto interakcií sa skrýva v tom, že rozhranie je založené na už existujúcich schopnostiach užívateľa.
- Priama interakcia znamená že medzi človekom ovládajúcim počítač a počítačom vlastným sa nenachádzajú ďalšie pomocné hardvérové zariadenia ako myš alebo klávesnica. Tak môže užívateľ ovládať systém s NUI rozhraním vlastnými prirodzenými akciami: hlasom alebo pohybmi tela. Tento prístup umožňuje, aby sme túto interakciu reálne prirovnali ku každodenným interakciám medzi človekom a svetom okolo neho. Prirodzenosť rozhrania je zabezpečená práve tým, že na ovládanie daného systému používa užívateľ vlastné telo – „You Are the Controller“ - „Ovládačom ste Vy“ (najznámejšie reklamné heslo Kinectu)[5]. V reálnom svete, keď sa napríklad chceme dotknúť obyčajnej stoličky, vo väčšine prípadov nepoužívame pomocné predmety, dotkneme sa jej priamo. NUI rozhranie realizuje túto priamosť aj pri komunikácii medzi človekom a počítačom.

Z analýzy definície uvedenej na začiatku tejto kapitoly sú zrejmé dve najdôležitejšie vlastnosti rozhrania NUI. Prvá taká vlastnosť je, že k ovládaniu systému sa využívajú schopnosti človeka, ktoré už vlastní, nie je potrebné, aby sa naučil niečo nové. Druhá vlastnosť je bezprostredná interakcia užívateľa a počítača. NUI využíva tieto svoje základné vlastnosti kombinovane: prirodzené prejavy človeka z jeho každodennej interakcie s okolitým svetom (pohyby, hlas) využíva na priamu

[5] Kinect Ads: 'You Are the Controller' (http://www.microsoft.com/en-us/news/features/2010/oct10/10-21kinectads.aspx).

komunikáciu s počítačom. Tento spôsob ovládania počítača je pre drvivú väčšinu užívateľov celkom jednoznačná a tak aj jednoduchá a pohodlná.

Ako som už v predchádzajúcich odsekoch naznačil, užívateľ ovláda NUI rozhranie vlastným hlasom a pohybmi tela. Tieto prejavy môžu byť veľmi rôzne, preto sa ich nedá rozpoznať a analyzovať len jedným rozhraním (rozpoznávanie reči a detekcia pohybu predstavujú dve veľmi odlišné oblasti informačných technológií).

Celú množinu NUI rozhraní môžeme rozdeliť na podmnožiny podľa typu vstupného dátového toku. Jeden z možných rozdelení podľa (7) je nasledujúce: rečové, dotykové a kinetické rozhranie. Toto rozdelenie je celkom pochopiteľné, keď si uvedomujeme aké veľké rozdiely sú v detekcii rôznych typu prejavov. Hlas sa rozpoznáva pomocou mikrofónu; dotykové rozhranie používa dotykovú obrazovku alebo touchpad; detekcia pohybu je možné kamerou (alebo kamerami), infračerveným senzorom alebo pohybovými senzormi.

V tejto práci ako vstupné zariadenie pre rozpoznávanie rôznych prejavov človeka používam senzor *Kinect*. Tento senzor má zabudovanú kameru, infračervený snímač a viac mikrofónov, a tak rozpozná hlas užívateľa aj jeho pohyby. Z vybavenia Kinectu vyplýva, že práca je zamierená na hlasové a kinetické NUI rozhranie. V nasledujúcich kapitolách sa zaoberám teoretickými základmi potrebnými k návrhu práve takýchto rozhraní.

2.2.1 Hlasové NUI rozhranie

Medziľudská komunikácia je založená hlavne na výmene informácií v podobe hlasov, jednoduchšie: na reči. Človek si pri komunikácii ani neuvedomuje, aký komplexný proces je rozpoznávanie reči, ľudský mozog našťastie zvláda túto úlohu bez akejkoľvek námahy. Rozpoznávanie reči počítačmi je už oveľa ťažšia úloha, presné určenie obsahu hlasových úsekov je veľmi náročné. Ľudský mozog, ktorý je potrebný k rozpoznaní reči, je v počítači nahradený procesorom a rôznymi algoritmami; namiesto ľudského ucha je vstupným zariadením mikrofón.

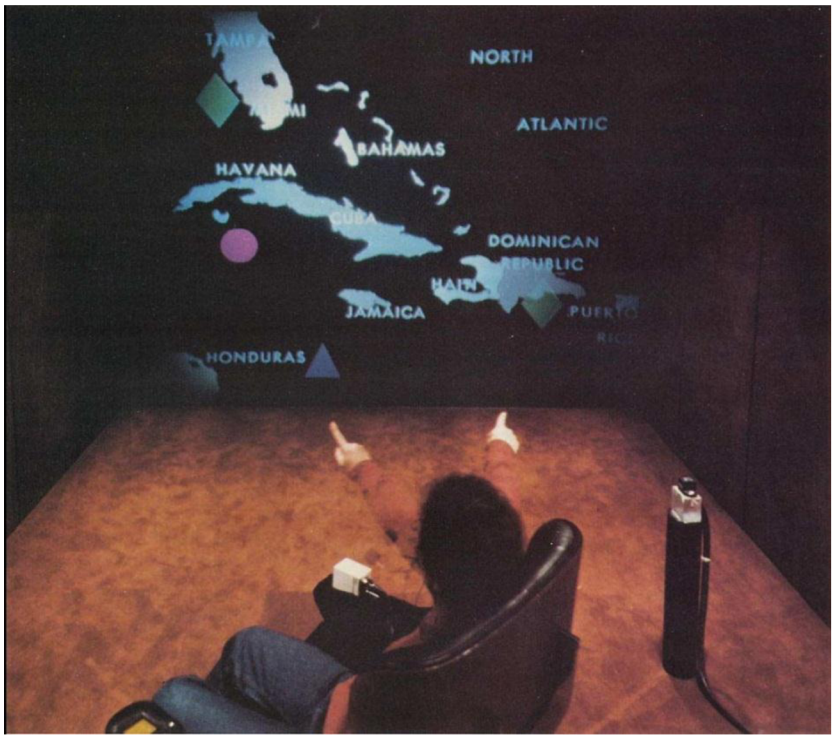
Rozhranie rozpoznávania hlasu môžeme rozdeliť do dvoch skupín: rozpoznávanie hlasových povelov a rozpoznávanie plynulej reči. Zložitosť rozhraní druhej skupiny je samozrejme vyššia ako u skupiny prvej.

U rozpoznávaní hlasových povelov nepotrebujeme rozpoznať každé slovo užívateľa, systém je zameraný na dopredu definované povely, alebo kľúčová slová, ktoré tvoria gramatiku rozpoznávača. Tieto povely a kľúčové slová je systém schopný vyfiltrovať aj z plynulej reči, a to nasledujúcim spôsobom: systém rozdelí plynulú reč na slová, a nasledovne ich porovná s kľúčovými slovami z preddefinovanej gramatiky. Ak sa v reči vyskytlo jedno alebo viac kľúčových slov, výstupom rozpoznávača bude rozpoznané kľúčové slovo. Na rozpoznávanie kľúčových slov sa často používajú skryté Markovské modely.

Rozpoznávanie plynulej reči otvorí veľa nových možností v komunikácii medzi človekom a počítačom. V budúcnosti nebudeme potrebovať vstupné zariadenia ako sú myš a klávesnica, na ovládanie počítača bude celkom dostačujúce sa s ním porozprávať. Napriek tomu, že s ich vývojom sa začalo už na konci osemdesiatych rokov dvadsiateho storočia, takéto rozhrania nájdeme dnes ešte len vo sci-fi filmoch alebo vo výskumných laboratóriách, Priekopníkmi hlasového rozhrania boli Chris Schmandt a Richard A. Bolt projektom *Put That There* (8)[6]. V rámci tohto projektu sa vytvorilo rozhranie, v ktorom môže užívateľ vytvoriť, premiestniť a vymazať objekty z premietanej obrazovky

[6] Put That There -Original video (http://www.youtube.com/watch?v=RyBEUyEtxQo).

vyslovením jednoduchých príkazov (napr.: „Vytvor modrý štvorec...tam.“ – na obrazovke sa objaví modrý štvorec na mieste kurzora).



Obrázok 2 Put That There (8)

V ďalších kapitolách tejto práce sa detailnejšie venujem prvému typu hlasového rozhrania: rozpoznaniu hlasových povelov.

2.2.2 Kinetické NUI rozhranie

Pri skúmaní medziľudských interakcií je jasné, že človek pri komunikácii používa okrem reči aj svoje telo. Reč tela je veľmi dôležitou súčasťou komunikácie, jedno gesto, alebo naše držanie tela o nás veľakrát prezradí viac, ako naše slová. Tieto drobné pohyby sú súčasťou nášho každodenného života. Kinetické NUI rozhranie môžeme ovládať práve s takými drobnými akciami nášho tela. Prirodzenosť rozhrania je založená na použití jednoduchých pohybov pre ovládanie nejakého systému bez nutnosti ďalších zariadení.

Kinetické prejavy použité pre ovládanie rozhraní môžeme podľa plynulosti triediť na statické a na dynamické:

- **Statické prejavy** predstavujú rôzne spôsoby držania tela. Do tejto podmnožiny prejavov patria akcie, pri ktorých skúmaná časť tela je v rozpoznateľnej polohe a pritom sa nehýbe. Je jedno, či na ovládanie použijeme len zápästie a polohu prstov, alebo celé telo vrátane nohy, kým dané časti tela sa nehýbu, rozprávame o statických prejavov.
- **Dynamické prejavy** sú rôzne pohyby, vykonané hocijakou časťou tela. Do tejto množiny patria všetky akcie od najmenších pohyboch prstov až do aktivít (beh, skákanie, apod.), pri ktorých sa človek hýbe celým telom.

Hore uvedené triedenie sa stane veľmi dôležitým pri návrhu a implementácii rozhrania pre detekciu ovládacích kinetických prejavov. Na rozpoznanie statických prejavov tela je celkom

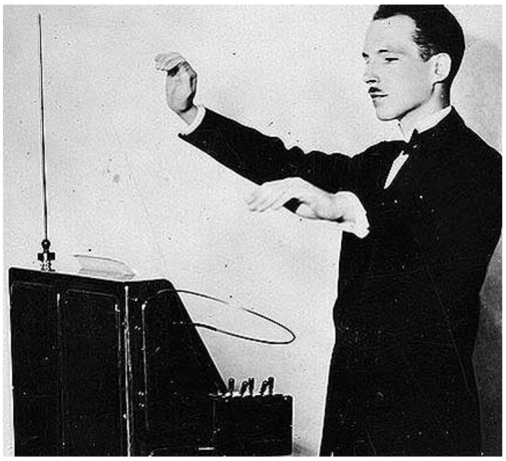
dostačujúce poznať aktuálne pozície skúmaných častí tela. U dynamických gest potrebuje rozhranie nielen aktuálnu polohu tela, ale aj predchádzajúce pozície jeho skúmaných častí.

Získať presnú aktuálnu polohu a vzájomnú polohu jednotlivých častí tela nie je vôbec jednoduchou úlohou. Medzi najčastejšie používanými nástrojmi na riešenie tejto úlohy patria kamery, infračervené snímače a rôzne kinetické senzory. Podľa spôsobu rozpoznania pohybu ich môžeme rozdeliť do dvoch skupín: senzory, ktoré snímajú priamo pohyb človeka; kamery a snímače, ktoré sú prostriedkami počítačového videnia. Zariadeniami, ktoré využívajú počítačové videnie, a metódami, ktoré sú na počítačovom videní založené sa zaoberám v nasledujúcich kapitolách.

Do prvej skupiny môžeme zaradiť tzv. *dátové rukavice*[7] („*cyberglove*" alebo „*wired glove*"), na ktorých sú umiestnené magnetické alebo zotrvačníkové sledovacie senzory, ktoré rozpoznajú pohyby prstov. Získané informácie je možné použiť pre detekciu rôznych dynamických gest. Existujú aj varianty s haptickou spätnou väzbou, ktoré slúžia aj ako výstupné zariadenia.

Obrázok 3 CyberGlove II (zdroj obrázku[8])

Ani do jednej uvedenej skupiny nemôžeme zaradiť menej známy elektronický hudobný nástroj s kinetickým rozhraním, zvaný *theremin*[9]. Pri hre na thereminu nie je vôbec nutné, aby sme sa dotýkali nástroja. Frekvenciu a amplitúdu výstupného zvukového signálu môžeme ovládať len pohybmi rúk. Charakteristiku hlasu thereminu ovplyvní len vzdialenosť ruky od kovových antén.



Obrázok 4 Leon Theremin pri hre na vlastnom nástroji (zdroj obrázku[10])

[7] CyberGlove Systems (http://www.cyberglovesystems.com/index.php).
[8] Zdroj obrázku: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/Realidad%20Virtual/web/imagenes/dispositivos/cyberglove.jpg
[9] What's a Theremin? (http://www.thereminworld.com/Article/14232/what-s-a-theremin-).
[10] Zdroj obrázku: http://www.freeinfosociety.com/article.php?id=441

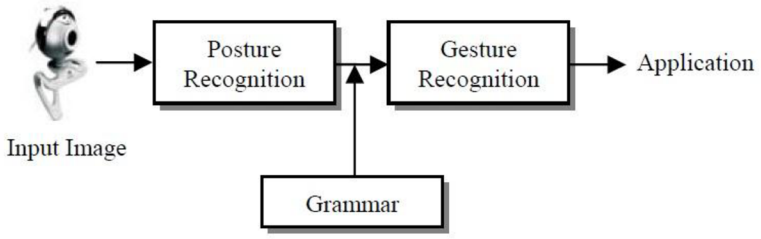
2.3 Rozpoznávanie pohybov človeka založené na počítačovom videní

Jedným spôsobom, ktorým je možné rozpoznať pohyby človeka v priestoru je používanie prvkov počítačového videnia. Tieto metódy sú založené na analýze snímok z kamier pomocou algoritmov pre spracovanie obrazu. Na začiatku tejto kapitoly predstavím zjednodušený koncept týchto metód a definujem pojmy gesto a póza používané už aj v predchádzajúcich častiach práce. V druhej časti kapitoly sa zaoberám hlavne s rôznymi technikami, ktoré slúžia k detekcii človeka v danom obraze a na rozpoznávanie dynamických gest a statických póz. Implementačná časť práce je založená na týchto metódach a technikách pracujúcich s farebnými a hĺbkovými snímkami.

2.3.1 Základné koncepty rozpoznávania pohybu v obraze

Snímky z kamier neobsahujú žiadne informácie o tom, kto, alebo čo na obrázku je a čo sa práve odohráva v skúmanej miestnosti. Z kamery dostaneme len postupnosť hodnôt reprezentujúcich jednotlivé pixely snímky. Aby rozpoznávanie pohybujúceho sa užívateľa bolo dosiahnuteľné, musíme najprv analyzovať každý pixel zvlášť a zistiť presnú polohu človeka v priestoru.

Po lokalizácii užívateľa v obrazu je nutné zistiť vzájomnú pozíciu častí jeho tela, čo nám vlastne udáva jeho držanie tela. Podľa týchto informácií už môžeme rozpoznať statické prejavy, o ktorých som sa zmienil v kapitole o kinetických NUI rozhraniach.



Obrázok 5 Metóda rozpoznávania dynamických gest (9)

Dynamické prejavy sú pohybmi človeka, preto ich nie je možné rozpoznať z jednej snímky. Najjednoduchším spôsobom rozpoznávania týchto pohybov môže byť nasledujúca metóda: z každej snímky zistíme spôsob držania tela užívateľa a túto informáciu uložíme do pamäti. Podľa aktuálnej snímky a predbežne uložených dát rozpoznáme aktuálne pohyby užívateľa.

Priestor v ktorom sa užívateľ pohybuje je trojdimenzionálna pravotočivá karteziánska súradnicová sústava, jej začiatok je na pozícii kamery/kamier. Pri pozorovaní užívateľa v priestoru je nutné rozhodnúť, aké informácie potrebujeme vedieť o jeho polohe. Na ovládanie jednoduchého NUI rozhrania je dostačujúce poznať horizontálne a vertikálne súradnice jednotlivých častí tela. Na zistenie týchto údajov nám stačí jedna obyčajná farebná kamera, potrebné hodnoty môžeme vypočítať pomocou aktuálnej snímky. Ak by naše rozhranie potrebovalo poznať aj hĺbku polohy užívateľa, to by už nebolo možné pri použití len jednej farebnej kamery. Hĺbka je vzdialenosť medzi kamerou a užívateľom, v hore predstavenej súradnicovej sústave súradnica *z*. Túto hodnotu nie je možné vypočítať z obrazového dátového prúdu jednej farebnej kamery, k výpočtu ich potrebujeme minimálne dve. Alternatívnou možnosťou pre zistenie informácií o hĺbke je používanie infračerveného senzoru, ktorý je schopný vytvoriť hĺbkovú mapu skúmaného priestoru.

Rozpoznávanie pohybujúceho sa užívateľa počítačovým videním a metódami spracovania obrazu je veľmi náročné. Aby výsledné rozhranie na ovládanie počítača podľa (10) bol spoľahlivý a prívetivý pre užívateľa, malo by splňovať nasledujúce požiadavky:

- **Robustnosť** – V reálnom svete môže byť vizuálna informácia nevyvážená, zašumená, neúplná. Tieto nedokonalosti sa môžu vyskytnúť pri zmenách osvetlenia, rušení, v prípade dynamického pozadia. Rozpoznávanie pohybu založené na počítačovom videní by malo byť nezávislé na užívateľovi a odolný proti vyššie zmieneným faktorom.
- **Výpočtová efektivita** – Ovládanie NUI rozhraní, ktoré sú založené na rozpoznávaní pohybu pomocou počítačového videnia beží v reálnom čase. Preto techniky a algoritmy pre rozpoznávanie by mali byť veľmi efektívne.
- **Tolerantnosť voči užívateľom** – Chyby užívateľa pri interakcii s rozhraním založeným na počítačovom videní by mali byť tolerované. Pri výskytu chyby na strane užívateľa, by ho rozhranie malo požiadať, aby svoje akcie zopakoval.
- **Rozšíriteľnosť** – Rozpoznávanie založené na počítačovom videní by malo byť prispôsobiteľné k rôznym typom aplikácií: desktopové a virtuálne prostredia, rozpoznávanie posunkovej reči, navigácia robotov.

Vedľa nedokonalostí snímok z kamery popísané u prvej požiadavky sa musíme zmieniť aj o ďalších ťažkostiach vyskytujúcich sa pri rozpoznávaní pohybu užívateľa. Pri ovládaní kinetického NUI rozhrania je vstupným zariadením sám užívateľ. Podľa (7) je táto skutočnosť nesmierne zaujímavá, ale prinesie so sebou aj nové problémy. Pohyby rôznych užívateľov pri vykonaní daného gesta pre ovládanie rozhrania nie sú úplne rovnaké, u každého užívateľa budú trošku iné. Nie je možné od užívateľa požadovať, aby napríklad rukou vykreslil kruh dokonale a totožne päťkrát za sebou. Aby kinetické rozhranie pracovalo správne, malo by byť odolné aj voči problémom vyššie popísaným, a malo by tolerovať nedokonalé statické a dynamické prejavy užívateľa.

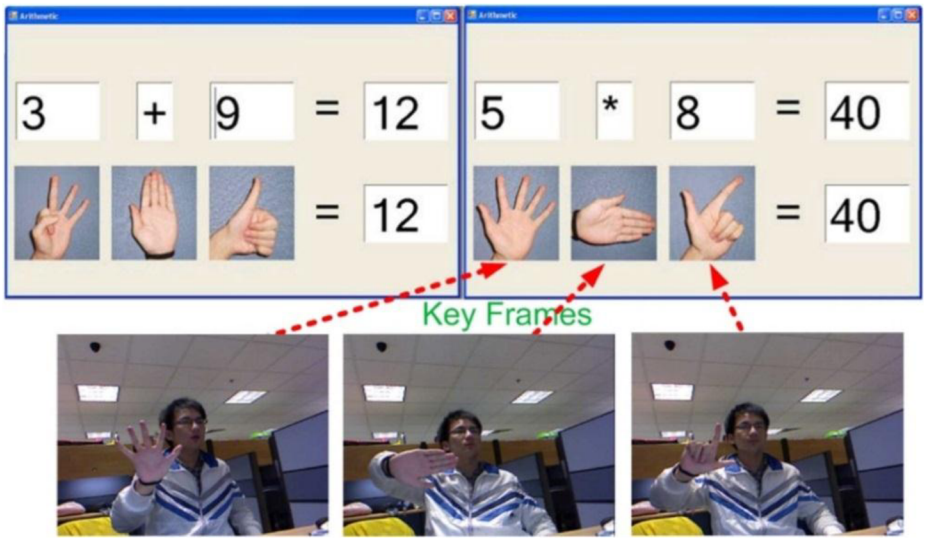
2.3.2 Rozpoznanie statických póz

V predchádzajúcej časti práce som sa už zaoberal popisom statických prejavov užívateľa, teraz na základe (7) definujem pojem *statická póza*.

Statická póza je druhom statických prejavov užívateľa slúžiacich na ovládanie kinetických NUI rozhraní založených na počítačovom videní, ktorú charakterizuje neprirodzené držanie tela.

Z definície plynie, že do skupiny póz patria také spôsoby držania tela, ktoré nie sú obvyklé pri každodenných pohyboch človeka. Hlavná myšlienka druhej časti definície sa vzťahuje na požiadavku jednoduchosti a prívetivosti kinetického NUI rozhrania pre užívateľa. Medzi spôsobmi držania tela s ktorými môžeme ovládať rozhranie by nemali patriť spôsoby, ktoré sú príliš obvyklé a každý deň ich často používame. V takom prípade by sme totiž mohli nechcene a omylom spustiť funkcie rozhrania. Dobrým príkladom je vynechanie takých držaní tela, ktoré používame pri odpočinku – napr. ruky za hlavou.

Na Obrázku 6 sú znázornené rôzne pózy ruky, ktoré sú používané na ovládanie kalkulátoru. Aplikácia rozpozná polohy prstov, ktorými môžeme zadať číslice a operátory. Aplikácia používa na rozpoznanie ruky senzor Kinect. (11)



Obrázok 6 Kalkulátor ovládaný ručnými pózami (11)

Ako som v predchádzajúcej kapitole už naznačil, rozpoznanie póz je omnoho jednoduchšie, ako rozpoznať dynamické gestá. Preto môžeme pri ovládaní rozhraní využívať pózy na často používané funkcie – efektivita rozhrania sa zvyšuje, rozpoznanie pózy je možné podľa aktuálnej snímky kamery, nie je potrebné poznať predchádzajúce stavy pohybu užívateľa.

Autori literatúr si často pletia pózy s gestami. Je dôležité si uvedomiť, že póza je statický spôsob držania tela; pri póze sa užívateľ nehýbe. Gesto je čistý pohyb; užívateľ ho vykoná rôznymi pohybmi častí svojho tela.

2.3.3 Rozpoznávanie dynamických gest

Gestá sú dynamickými prejavmi užívateľa pri ovládaní kinetického NUI rozhrania. Popis takýchto prejavov je veľmi všeobecný, je nutné, aby sme gestá definovali trošku konkrétnejším spôsobom:

„Gesto je pohyb tela, ktorý prenáša informácie. Mávanie na rozlúčku je gesto. Stlačenie klávesy na klávesnici nie je gestom, pretože pohyb prstov pri stlačení klávesy nie je skúmaný a nie je významný. Dôležitý je, ktorý kláves bol stlačený.“ (12)

Gesto je pohyb užívateľa, kto chce ovládať počítač pomocou kinetického NUI rozhrania. Tento pohyb musí obsahovať informácie pre rozhranie, popisujúce jeho požadované správanie. Podľa týchto informácii rozhoduje rozhranie o tom, čo vlastne užívateľ chcel dosiahnuť vykonaním rozpoznaného gesta. Táto informácia je významná len pre dané kinetické rozhranie, lebo gesto nosí špecifickú informáciu, ktorý má význam len pre neho.

Definícia jasne udáva a príkladom aj znázorňuje ktoré pohyby sú gestá a aké činnosti užívateľa medzi nimi nepatria. Pred návrhom kinetického rozhrania ovládaného gestami je veľmi dôležité, aby sme určili rozdiel medzi gestom a pohybom, ktorý nesplňuje požiadavky definície (pohyby, ktoré tieto požiadavky nesplňujú nazývame *manipulácie*).

Prvá časť definície tvrdí, že gesto vždy prenáša nejakú informáciu. Manipulácie takú informáciu neobsahujú. Aby sme sa držali uvedeného príkladu z definície: samotný pohyb, ktorý vykonáme pri stlačení klávesy pre nás žiadny význam nemá, kým nevieme, ktorý kláves bol stlačený. Je jedno akým spôsobom a akými pohybmi je ten kláves stlačený, výsledok akcie bude stále stlačenie totožného klávesu.

Keby sme chceli sumarizovať odlišnosti medzi gestami a manipuláciami, dospeli by sme k tomu, že:

- u manipuláciách má význam len výsledok akcie, kým
- u gest výsledná pozícia skúmanej časti tela nie dôležitá. Významná informácia je ukrytá v priebehu pohybu užívateľa predstavujúce dané gesto.

Porozumenie definícii gesta sa stane veľmi dôležitým pri návrhu kinetického NUI rozhrania. Návrh nášho systému by mal z rozhrania úplne vypustiť ovládanie pomocou manipulácií.

Rozpoznávanie gest tela alebo ruky môže byť veľmi jednoduchá aj extrémne komplexná. Zložitosť detekcie je v každom prípade závislý na aktuálnom gestu, ktoré chceme rozpoznať. V tejto časti kapitoly sa zaoberám s metódami, ktoré sú použiteľné na rozpoznanie všeobecných gest.

Rozpoznávacie metódy podľa (7) je možné rozdeliť do troch skupín:

1. **Rozpoznanie pomocou algoritmov**: Táto metóda je najjednoduchšia a najpriamočiarejšia zo všetkých troch. Je založená na definovaní rôznych pravidiel a podmienok, ktoré sú následne aplikované na pohyby užívateľa. Výsledkom algoritmického rozpoznávania je vykonané gesto, alebo odmietnutie gesta podľa niektorého pravidla alebo niektorej podmienky. Kvôli svojej jednoduchosti, algoritmická metóda má ťažkosti rozpoznať komplexnejšie gestá. Algoritmami definovať zložitejšie gesto, ako je napríklad beh alebo hod, je veľmi ťažká úloha. V najčastejších prípadoch takéto zložité algoritmy nie sú dostatočne robustné, aby boli použiteľné v reálnych aplikáciách. Ďalšou nevýhodou algoritmického rozpoznania je nerozšíriteľnosť, jedným algoritmom môžeme popísať len jedno gesto. Práve preto môže rozhranie detekujúce viaceré gestá touto metódou byť veľmi neefektívne a náročné na výpočtové zdroje.

   Metóda algoritmického rozpoznávania je efektívna v prípade použitia menšieho počtu jednoduchých gest.
2. **Rozpoznávanie pomocou neurónových sietí**: Definovať komplikovanejšie gestá pomocou algoritmov je vo väčšine prípadov skoro nemožné. Algoritmy sú príliš jednoduché a rozpoznávanie ich pomocou nie je dostatočne robustný. Detekcia neurónovými sieťami je založená na vyhodnotení štatistických údajov a pravdepodobnosti. Výsledok rozpoznávania je pravdepodobnosť, že užívateľove pohyby predstavovali niektoré známe gesto. Táto metóda rieši aj problém rozšíriteľnosti. Neurónová sieť pozostáva z umelých neurónov, ktoré obsahujú jednoduché algoritmy a spracovávajú menšie časti gesta. Jeden neurón môže byť využívaný pri rozpoznávaní viacerých rôznych gest, ale rozpoznávanie žiadnych dvoch gest nevyužíva tie isté neuróny. Nevýhodou tejto metódy je vyššia komplexnosť voči algoritmickému prístupu. Zostaviť neurónovú sieť pre rozpoznanie gest je veľmi náročné.

   Použiť metódu s neurónovými sieťami sa oplatí v prípade, že chceme rozpoznať zložité gestá, ktoré sa nedajú presne definovať algoritmami.
3. **Rozpoznávanie príkladom**: Táto metóda je založená na porovnaní pohybov užívateľa so známymi formami daných gest pri využívaní strojového učenia. K detekcii nevyhnutne potrebujeme databázy s vykonanými gestami od rôznych ľudí. Aby úspešnosť rozpoznania bola vysoká, systém musí poznať gestá vykonané rôznymi spôsobmi. V prípade veľkej databázy bude úspešnosť vysoká ale náročnosť systému na výpočtové zdroje tiež rastie.

Metóda rozpoznávania príkladom je najúspešnejšia z troch metód a pridanie nového gesta je veľmi jednoduché. Nevýhodou takýchto systémov je ale potreba veľkej databáze a v mnohých prípadoch vyššie nároky na výpočtové zdroje.

V tejto podkapitole som predstavil tri všeobecné metódy, ktoré sú použiteľné na rozpoznávanie dynamických gest slúžiace pre ovládanie kinetických NUI rozhraní. S prvou metódou sa zaoberám detailnejšie v nasledujúcej časti tejto práce.

Prehľad techník na rozpoznávanie gest ruky:

Predchádzajúce odseky popisujú základné metódy na rozpoznávanie všeobecných gest. Na ovládanie kinetického NUI rozhrania používame najčastejšie naše ruky. Rozhranie navrhované v tejto práci je tiež založené na rozpoznávaní gest rúk. Kvôli tomu tu, na konci kapitoly zaoberajúcej sa prirodzenými užívateľskými rozhraniami uvádzam nasledujúci krátky prehľad techník na rozpoznávanie gest ruky podľa (13).

Systémy ovládané gestami ruky sú obvykle založené na trojvrstvovom modelu. Tieto tri vrstvy sú nasledujúce: *detekčná vrstva* sa zaoberá extrakciou charakteristických vlastností z obrazu, a detekciou ruky; *sledovacia vrstva* spája informácie zo snímok idúcich za sebou; *rozpoznávacia vrstva* pracuje s dátami z predchádzajúcich dvoch vrstiev a podľa extrahovaných dát rozpoznáva či užívateľ vykonal hľadané gesto alebo nie.

1. **Detekčná vrstva**: hlavná úloha tejto vrstvy je detekcia ruky a segmentácia hľadanej časti snímky. Táto segmentácia je veľmi dôležitá, pretože oddelí dôležité dáta od pozadia obrazu. Po rozčlenení obrazu posiela vyextrahované dáta k sledovacej a rozpoznávacej vrstve. Na segmentáciu významných oblastí obrazu môžeme použiť nasledujúce metódy:
    - **Farba** - segmentácia podľa farby ruky. Najčastejšie používanými farebnými systémami na túto úlohu sú *HSV*, *YCrCb*, *RGB* a *YUV*. Dôležité je dosiahnutie istého stupňa robustnosti proti zmenám osvetlenia ruky.
    - **Tvar** – ruka má charakteristický tvar, detekciu neovplyvní farba kože a zmena osvetlenia. Extrahovaním kontúr objektov môžeme získať použiteľné informácie. Nevýhodou tohto prístupu je potreba post-processingu pre zvýšenie spoľahlivosti segmentácie.
    - **Boosting** – aplikácia princípov strojového učenia – *AdaBoost*, *FloatBoost*.
    - **Detekcia pomocou 3D modelu** – táto metóda používa 3D model ruky na detekciu ruky na obraze. Výhodou týchto metód je ich nezávislosť od polohy a pozícii ruky.
    - **Detekcia ruky podľa pohybu** – predpokladom detekcie je statické pozadie, niektorých z metód predpokladajú, že na obraze sa pohybuje len ruka.
2. **Sledovacia vrstva**: vrstva spojí segmentované oblasti ruky alebo extrahované charakteristické javy. Týmto spôsobom je možné zistiť dráhu ruky v čase.
    - **Sledovanie na základe vzoru** – detekcia ruky sa odohráva len okolo polohy ruky z predchádzajúcej snímky. Pre optimálne fungovanie potrebuje táto metóda istý počet snímok za sekundu.
    - **Techniky optimálneho odhadu** – tieto techniky používajú filtrovanie obrazu *Kalman filtrom*.
    - **Sledovanie založené na algoritmu Mean Shift** a **Particle Filtering**

3. **Rozpoznávacia vrstva**: Táto časť systému interpretuje dáta a informácie o polohe ruky získané v predchádzajúcich vrstvách.
    - **Porovnanie vzorov** – použiteľná na rozpoznanie póz a gest. Porovnávanie snímok sa odohráva na úrovni pixelov.
    - **Metódy založené na PCA** – najčastejšie používané na detekciu póz algoritmom *PCA* (*Principal Component Analysis*).
    - **Boosting** – metódy, ktoré je možné použiť na detekciu ruky v prvej vrstve, sú použiteľné aj na rozpoznanie póz.
    - **Porovnávanie kontúr a siluety ruky** – Metódy sú schopné rozpoznať statické pózy porovnaním obrazu ruky s prototypovými obrazmi.

3 Senzor Kinect

Prirodzené užívateľské rozhrania sú založené na sledovaní prejavov užívateľa. Ovládať takéto rozhrania môžeme priamo naším hlasom alebo rôznymi pohybmi nášho tela. Prirodzenosť rozhrania znamená, že užívateľ komunikuje priamo s rozhraním, nie je potrebné použitie ďalších pomocných hardvérových nástrojov. Jedná sa síce o priamu interakciu medzi užívateľom a počítačom, aj toto rozhranie potrebuje určitý typ vstupného zariadenia na pozorovanie užívateľa. Tieto zariadenia môžu byť bežne používané hardvéry ako sú mikrofón a webová kamera, pozorovať užívateľa je možné aj s takýmito obvyklými vstupnými nástrojmi počítača.

Vývoj v oblasti informačných technológií *interakcia človek – počítač* (HCI) je od začiatku nového tisícročia zameraný hlavne na aplikáciu prvkov prirodzených užívateľských rozhraní v praxi. Vďaka tomuto vývoju sa na trhu objavili nové produkty ako *PlayStation EyeToy* a *Eye*, *Microsoft Kinect* a spolu s chytrým telefónom *Apple iPhone* technológia viacdotykových obrazoviek (ovládače *PlayStation Move* a *Wii Remote* nesplňujú požiadavky prirodzeného užívateľského rozhrania, lebo ovládanie herných konzol s týmito zariadeniami nie je priame, direktné – užívateľ drží tieto ovládače v ruke, konzola sa ovláda pomocou informácií o pozícii a zrýchlenia ovládača). Zmienené nové vstupné zariadenia sú špecializované hardvéry, ktoré boli vyvinuté na sledovanie užívateľa v priestoru a na ovládanie herných konzol.

Výhody týchto špecializovaných zariadení slúžiacich na pozorovanie užívateľa oproti obvyklými kamerami a mikrofónmi sú jednoznačné. Špecializované senzory obsahujú hardvérové implementácie funkcií pre sledovanie a rozpoznávanie človeka. V prípade použitia obyčajných kamier, programátor musí tieto funkcie implementovať v softvéru. Hardvérová implementácia nejakého algoritmu je v každom prípade rýchlejšia ako softvérové riešenie. Keď si uvedomíme, že algoritmy počítačového videnia a spracovania obrazu musia bežať v reálnom čase a niektoré z nich sú aj výpočtovo náročné, hardvérová implementácia rozpoznania človeka je veľkou pomocou pri návrhu a implementácii prirodzených rozhraní. V tejto práci požívam jeden z takých špecializovaných vstupných zariadení, konkrétne senzor *Kinect* od firmy *Microsoft*.

Táto kapitola predstavuje senzor Kinect, spôsob jeho fungovania a popíše históriu jeho vývoja. V druhej časti kapitoly sa zaoberám existujúcimi nástrojmi a knižnicami na implementáciu aplikácií pre senzor Kinect.

3.1 Komponenty senzoru Kinect

Senzor Kinect je vstupné zariadenie pôvodne dostupné od novembra 2010 k hernej konzole firmy Microsoft *Xbox 360*. Kinect je vstupné zariadenie hernej konzoly, ktoré rozpoznáva pohyby užívateľa. Pomocou Kinectu môžu užívatelia ovládať aplikácie a hry bežiace na Xbox-u 360 bez použitia hardvérového ovládača.

Obrázok 7 Microsoft Kinect for Xbox 360[11]

[11] Zdroj obrázku: http://msdn.microsoft.com/en-us/library/hh438998.aspx.

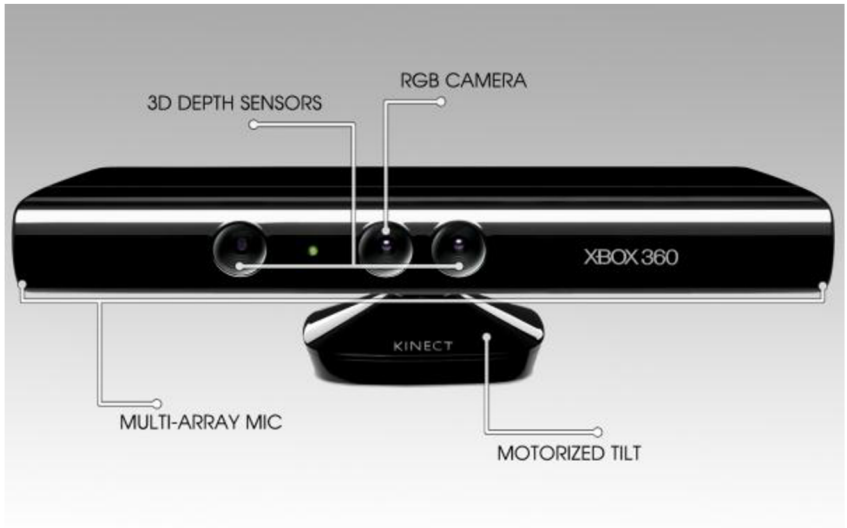
Podľa pôvodných plánov Microsoftu by toto zariadenie malo byť použiteľné iba s konzolou Xbox. Bezprostredne po termínu vydania sa ale objavili prvé informácie o „hacknutí“ senzoru – trvalo približne sedem dní, kým komunita programátorov bola schopná získať farebný a hĺbkový obraz z Kinectu pomocou vlastných ovládačov.

K „hacknutiu“ Kinectu napomáhala aj skutočnosť, že na pripojenie ku konzoly Xbox používa senzor bežný USB konektor. Pripojenie senzoru k počítači je preto veľmi jednoduché, nepotrebujeme k tomu žiadne ďalšie káble alebo adaptéry. Predchádzajúce tvrdenie je pravdivé v prípade, že sme senzor kúpili samostatne, nie spoločne s Xbox-om. V prípade, že si senzor kúpime spolu s Xbox-om, k pripojeniu Kinectu k počítači potrebujeme dokúpiť AC adaptér (na Obrázku 8) so špeciálnym konektorom ku Kinectu, ktorý nie je súčasťou balenia Xbox-u. Tento AC adaptér ale bohužiaľ nie je v tuzemských internetových a kamenných obchodoch bežne dostupný, dá sa zakúpiť v oficiálnom obchode Microsoft Store.

Obrázok 8 AC adaptér potrebný k pripojeniu Kinectu k počítaču (15)

Problém pripojenia Kinectu k počítači je už vyriešené, môžeme preskúmať, čo je do tohto senzoru vlastne zabudované. „Hlava“ Kinectu so senzormi je osadená na podstavci, ktorý má skoro štvorcový tvar. Základ skrýva malý elektromotor, pomocou ktorého môžeme hlavu senzoru nakloniť po vertikálnej ose podľa našej potreby. Uhol naklonenia môže byť maximálne dvadsaťsedem stupňov od vodorovnej polohy senzoru v oboch smeroch. Túto funkciu by sme mali využívať čo najmenej, pretože zabudovaný elektromotor nebol navrhnutý na časté použitie. Minimalizácia záťaže motoru je veľmi dôležitá; pri nesprávnom zaobchádzaní, môže dôjsť aj k jeho poškodeniu. Na zníženie záťaže motoru je polohovanie hlavy senzoru limitované maximálnou hodnotou: uhol je možné zmeniť len raz za sekundu; po pätnástich zmien idúcich za sebou musí elektromotor „odpočívať“ po dobu dvadsiatich sekúnd. V tomto intervalu nie je možné senzor nakloniť. (14)

Na hornej časti Kinectu sa nachádzajú senzory, pomocou ktorých sleduje užívateľa v priestoru: farebná kamera, infračervený projektor, infračervená kamera a štyri mikrofóny. Okrem týchto nástrojov nájdeme na Kinectu ešte aj LED indikátor stavu zariadenia.



Obrázok 9 Umiestnenie jednotlivých senzorov na Kinectu[12]

[12] Zdroj obrázku: http://entreprene.us/wp-content/uploads/2011/03/kinect_Hacks_Introduction.jpg.

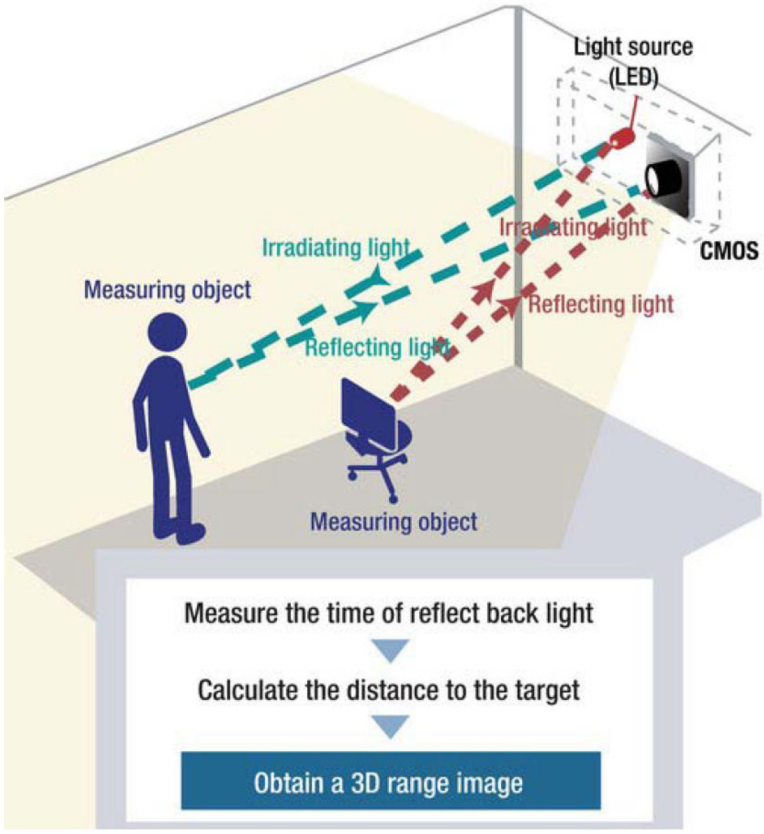
Obraz kamier má implicitné rozlíšenie 640x480 pri 30 snímkach za sekundu. Farebná kamera používa osem bitov na farebnú hĺbku, a na výstupnú snímku sa aplikuje Bayerov filter. Monochromatický obraz z infračervenej kamery má bitovú hĺbku 11 bitov.[13]

Kinect disponuje štyrmi mikrofónmi: jeden z nich sa nachádza na ľavej strane senzoru, ostatné sú umiestnené na pravý kraj zariadenia. Mikrofóny sú prepojené s 24-bitovým A/D prevodníkom a ďalšími obvodmi pre spracovanie signálu. Kinect má zabudovanú funkciu na potlačenie šumu a na zrušenie akustických ozvien.[14]

3.2 Structured Light kamerové systémy

Táto podkapitola bola prevzatá z (15).

Pre ovládanie prirodzeného užívateľského rozhrania potrebujeme sledovať užívateľa v priestoru a rozpoznať jeho pohyby. Na výpočet vzdialenosti medzi senzorom a skúmaným objektom potrebujeme minimálne dve farebné kamery, alebo špecializované zariadenie, akým je aj Kinect. Dneska už existuje mnoho typov takýchto zariadení, ich spoločnou charakteristikou je forma výstupu: senzory generujú monochromatickú hĺbkovú mapu alebo bodovú reprezentáciu trojdimenzionálneho priestoru. Základná technika senzorov na výpočet hĺbkových informácii je v týchto zariadeniach nasledovná: projektor vysiela signál, ktorý sa odrazí od skúmaného objektu v priestoru, senzor zachytí odrazený signál a podľa charakteristiky zachyteného odrazeného signálu vypočíta hĺbkovú informáciu.



Obrázok 10 Výpočet hĺbkovej informácie skúmaných objektov v priestoru (15)

[13] Kinect (http://en.wikipedia.org/wiki/Kinect).
[14] MSDN Library, Kinect Sensor (http://msdn.microsoft.com/en-us/library/hh855355.aspx).

Metóda, ktorá sa na výpočet hĺbkových informácií používa v senzoru Kinect je tzv. *Structured Light* metóda (ďalej metóda štruktúrovaného svetla). Táto metóda generuje hĺbkovú mapu pomocou projektovania štruktúrovaného svetla, ako sú napríklad frekvenčné svetelné pásy, kódované svetlo alebo rôzne vzory tvarov. Tieto vzory sa skresľujú a deformujú na objektoch skúmaného priestoru. Hĺbková vzdialenosť daného objektu je vypočítateľná podľa skreslenia a deformácie vzorov. Najčastejšie používaným svetelným signálom u metódy štruktúrovaného svetla je vzor horizontálnych pásov predstavený viditeľnou časťou spektra elektromagnetických signálov – horizontálne svetelné pásy.

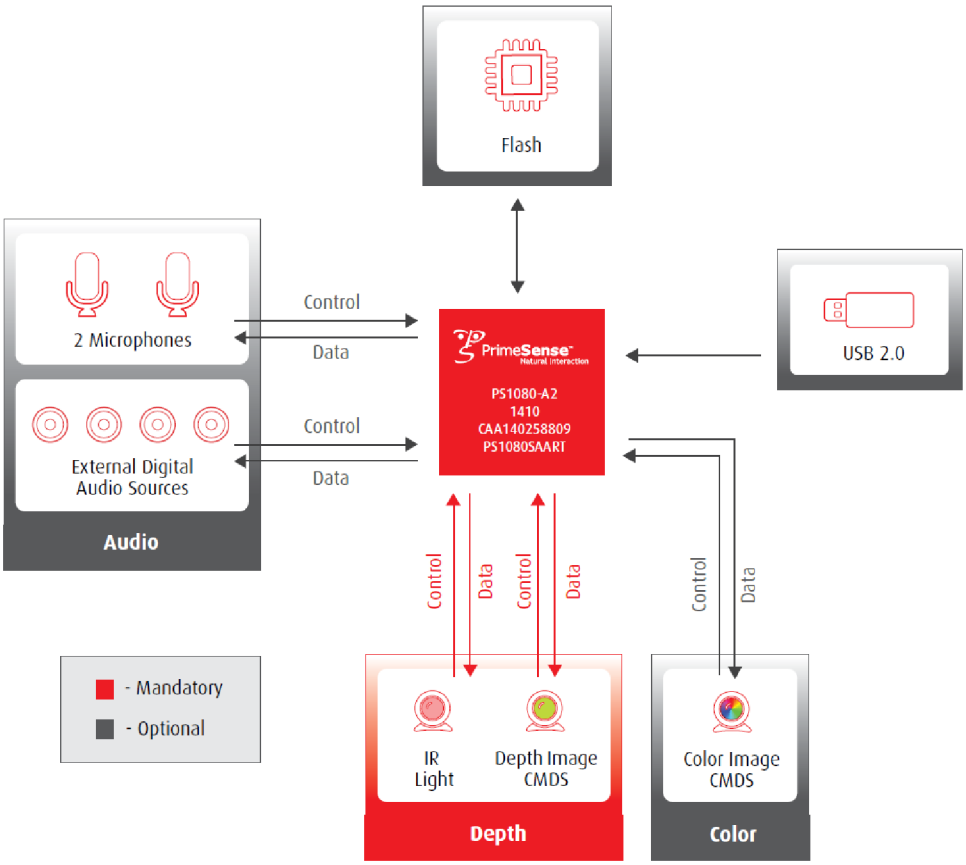
Používanie viditeľného štruktúrovaného svetla pre generovanie hĺbkovej mapy by u senzoru, aký je Kinect, bolo pre užívateľa veľmi rušivé. Preto hĺbkový senzor v Kinectu pracuje v neviditeľnej infračervenej časti spektra. Zabudovaný infračervený laser vrhá do priestoru statické body infračerveného svetla s rôznou intenzitou pomocou optickej mriežky. Na prvý pohľad je rozloženie týchto statických bodov náhodné, ale po podrobnom preskúmaní podľa (16) je jasné, že polohy bodov sú rozložené podľa vzoru.

Tieto statické body vrhané infračerveným laserom nie možné zachytiť obyčajnou farebnou kamerou. Preto je do Kinectu zabudovaná špeciálna kamera, ktorá „vidí“ infračervené žiarenie. Táto kamera má na svojej optike inštalovaný filter, ktorý vyfiltruje viditeľnú časť spektra. Kvôli tomu sa ku svetelnému senzoru dostanú len infračervené signály.

Výpočet hĺbkovej mapy podľa skreslenia zachytených infračervených bodov vykonáva zabudovaný čip *PS1080 SoC* od firmy *PrimeSense*. Okrem obvodov na spracovanie hĺbkových obrazových dát obsahuje tento čip, ešte aj všetky riadiace jednotky zariadenia: ovláda mikrofóny, farebnú kameru a USB rozhranie.

Vyššie popísanú štruktúru - *referenčný dizajn PrimeSense*, aj technológiu, ktorá je používaná na výpočet hĺbkovej mapy - Light Coding™ vlastní firma PrimeSense. Po zakúpení licencie Microsoft vytvoril Kinect, ktorý je vlastne prvým komerčným produktom, ktorý je založený na tomto referenčnom dizajnu (hardvérová štruktúra senzoru PrimeSense je na Obrázku 11).

Po úspechu Kinectu si viaceré firmy uvedomili výhody a možnosti hardvérového dizajnu od PrimeSense. Na trhu sa objavili ďalšie licencované produkty, ako napríklad *Xtion PRO* a *Xtion PRO Live* od firmy *Asus*.



Obrázok 11 Referenčný dizajn PrimeSense senzoru[15]

3.3 Ovládače a vývojové nástroje ku Kinectu

Na začiatku kapitoly som písal o jednoduchosti fyzického pripojenia Kinectu k počítači pomocou USB portu. Je síce pravda, že k pripojeniu nepotrebujeme ďalšie špeciálne adaptéry, ale bez vhodných ovládačov nemôžeme s Kinectom pracovať. Kinect je pôvodne príslušenstvom k hernej konzole Xbox 360, Microsoft neplánoval vydanie počítačovej verzie senzoru.

Kinect bol jedným z najočakávanejších noviniek na trhu hardvérov v roku 2010. Okrem hráčov a milovníkov konzoly Xbox 360, túto novinku čakala aj veľká skupina nadšených programátorov, ktorí chceli využiť možnosti Kinectu aj na osobnom počítači. Bez oficiálnych ovládačov od Microsoftu operačné systémy nemohli rozpoznať nové zariadenie, programátori nemali prístup k obrazovým výstupom a funkciám senzoru – taká bola situácia 4. novembra 2010, bezprostredne po vydaní Kinectu.

Firma *AdaFruit Industries* vyhlásila súťaž na implementáciu open-source ovládača, pomocou ktorej by funkcie a obrazové prúdy boli získateľné zo senzoru. Programátori začali pracovať na ovládači, objavili sa rôzne skupiny nadšencov, aká je napríklad komunita *OpenKinect*. Táto súťaž netrvala dlho, po siedmych dňoch sa objavil prvý ovládač na platformu Linux, ktorý bol schopný získať farebný a hĺbkový obraz z Kinectu. Po úspechu sa autor ovládača pripojil ku komunite OpenKinect, a spoločne vydali prvú sadu knižníc ku Kinectu – *libfreenect*. (7)

[15] Zdroj obrázku: PrimeSense 3D Sensor Data Sheet (http://www.primesense.com/en/press-room/resources/file/4-primesense-3d-sensor-data-sheet)

Ďalšou veľmi dôležitou udalosťou v histórii Kinectu z hľadiska jeho používania spolu s počítačom bolo založenie organizácie *OpenNI* firmou PrimeSense a vydanie open-source ovládača k senzoru. Okrem toho vydali ešte framework *NITE*, ktorý bol schopný rozpoznať kostru človeka.

Firma Microsoft, ktorá vyvinula senzor Kinect, po týchto udalostiach ohlásil beta verziu vlastného vývojového nástroja k senzoru. *Kinect SDK* bol voľne dostupný od júna 2011, jeho využitie bolo obmedzené na nekomerčné účely. Preto pre profesionálov slúžila táto verzia softvéru len ako testovací nástroj funkcií Kinectu, nadšenci programovania dostali robustnú sadu knižníc s rozpoznávaním reči a kostry človeka bez potreby kalibrácie. Výslednú komerčnú verziu nástroja vydal Microsoft pod názvom *Kinect for Windows SDK* spolu s novou verziou Kinectu, *Kinect for Windows*.

V nasledujúcej časti podkapitoly predstavím ovládače vývojových nástrojov ku Kinectu, ktoré boli vytvorené v rámci komerčných a open-source firemných projektov, alebo projektov komunít.

3.3.1 OpenKinect a libfreenect

Komunita OpenKinect (15) sa zaoberá vývojom open-source knižníc, ktoré umožnia prácu so senzorom Kinect na platformách Windows, Linux, Mac. Do tvorby projektu *libfreenect* sa zapojilo viac ako dvetisíc programátorov. Tento softvér bol prvým voľne dostupným nástrojom, ktorý umožnil získať farebné a hĺbkové snímky z Kinectu.

Knižnica libfreenect zaisťuje prístup k trom hlavným obrazovým výstupom Kinectu:

- Hĺbková mapa
- Infračervený obraz priamo z hĺbkovej kamery
- Obraz farebnej kamery

Ovládač knižnice libfreenect umožňuje nízkoúrovňovú prácu s Kinectom, a zabezpečí prístup k obrazu priamo z infračervenej kamery. Kombináciou týchto obrazových dát a hĺbkovej mapy je možné vykresliť 3D reprezentáciu skúmaného priestoru.

So softvérom komunity OpenKinect môžeme ešte ovládať funkciu naklonenia hornej časti Kinectu, a máme aj prístup k stavom zabudovanej stavovej LED diódy.

3.3.2 CL NUI Platform

AlexP bol prvý programátor, kto po vydaní Kinectu úspešne získal farebný a hĺbkový obraz zo svetelných senzorov zariadenia. Projekt CL NUI Platform (15) je založený na implementácii tohto programátora.

CL NUI Platform sa objavil v decembri 2010. Obsahuje SDK, API, ovládač podporujúci operačné systémy Windows a vzorovú aplikáciu v jazyku C/C++ a C#. Platforma podporovala pripojenie viacerých senzorov k počítači, pracovala s rôznymi formátmi obrazu farebnej a hĺbkovej kamery pri 30 snímkach za sekundu. Umožňovala ešte ovládanie malého elektromotora slúžiaceho k polohovaniu hlavy senzora a stavovej LED diódy.

3.3.3 OpenNI a OpenNI framework

Organizácia OpenNI (*Open Natural Interaction*) (17) bola založená v novembri 2010, po vydaní Kinectu. Medzi zakladateľov patria firmy *PrimeSense* (referenčný dizajn PrimeSense – základná štruktúra Kinectu), *Willow Garage* (skúsenosti v oblasti osobných robotov), *Side-Kick* (vývoj hier ovládaných pohybmi), *Asus* (svetoznáma výrobca hardvéru – Asus Xtion PRO a PRO Live senzory

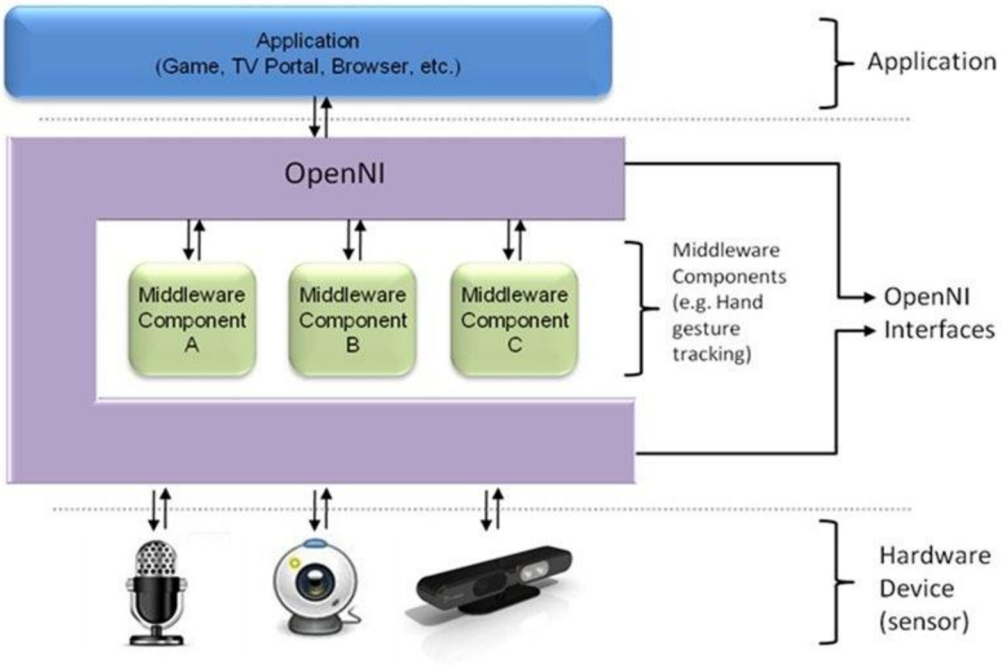
na základe referenčného dizajnu PrimeSense) a *AppSide* (prvý internetový obchod aplikácií pre zariadenia kontrolovaných pohybom). Hlavným cieľom organizácie je promócia prenositeľnosti aplikácií medzi rôznymi NI zariadeniami pomocou middleware[16] softvéru (NI - prirodzená interakcia).

OpenNI je open-source viacjazyčný framework prenositeľný medzi platformami, ktorý slúži na vytvorenie aplikácií s prirodzeným užívateľským rozhraním. Hlavným účelom frameworku je formulácia štandardizovaného API rozhrania, ktoré umožňuje komunikáciu s:

- zvukovými a svetelnými senzormi,
- strednou aplikačnou vrstvou medzi senzormi a aplikáciami, ktorá analyzuje signály zo zvukových a svetelných senzorov (komponenty softvérového middleware-u).

Štruktúra vyššie popísanej koncepcie je znázornená na *obrázku 12*. Je to vrstvový model s tromi vrstvami:

- Horná vrstva predstavuje aplikácie využívajúce programovacie rozhrania OpenNI.
- Dolná vrstva je skupina rôznych hlasových a svetelných senzorov.
- Stredná vrstva, ktorá spája aplikácie so senzormi, je vlastne OpenNI. Framework obsahuje rozhrania, ktoré komunikujú so senzormi, aj s komponentmi middleware-u, ktoré analyzujú signály zo senzorov.



Obrázok 12 Vrstvový model štruktúry OpenNI (17)

Framework OpenNI obsahuje také API[17] rozhrania, ktoré by mali byť implementované komponentmi middleware-u a ktoré by mali byť implementované senzormi. Táto štruktúra je veľmi výhodná z hľadiska prenositeľnosti aplikácií, komponentov middleware-u a senzorov. OpenNI umožňuje, aby

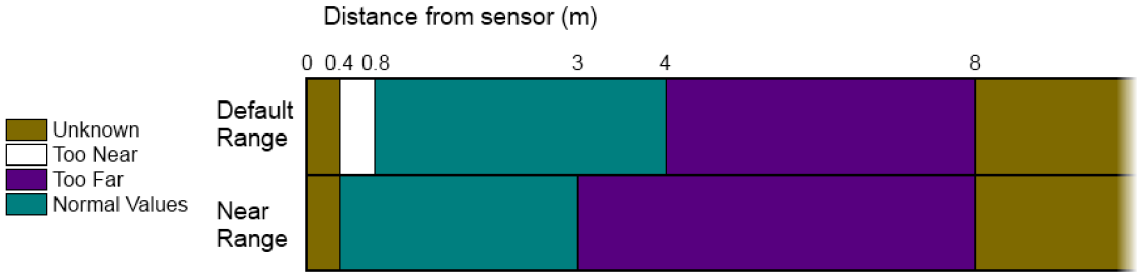
- aplikácie bežali nad rôznymi middleware modulmi,
- bolo jedno z akého senzoru dostanú komponenty middleware-u dáta daného formátu,
- výrobcovia senzorových zariadení vytvorili senzory, ktoré sú kompatibilné s aplikáciami využívajúcich OpenNI framework.

[16] Middleware: <http://en.wikipedia.org/wiki/Middleware>.
[17] Application Programming Interface: <http://en.wikipedia.org/wiki/Application_programming_interface>.

3.4 Kinect for Windows SDK

Výslednú verziu oficiálneho vývojového nástroja Kinect for Windows SDK vydal Microsoft vo februáru tohto roka. Nástroj obsahuje knižnice pre vývoj aplikácií pracujúcich s Kinectom, dokumentáciu týchto knižníc, ovládač k senzoru a ukážkové aplikácie so zdrojovými kódmi. Spolu s týmto nástrojom vyšla aj nová verzia senzoru Kinect for Windows, ktorá už je určená na pripojenie k osobným počítačom s operačným systémom Windows. Hardvér senzoru sa nezmenil, nová verzia len dostala firmware s novými funkciami: k jednému počítači môžeme pripojiť až štyri Kinecty, ďalej tzv. *Near Mode* – blízky režim, hĺbková kamera vidí aj bližšie objekty.

Pomocou novej funkcie blízkeho režimu vidí hĺbková kamera objekty už vo vzdialenosti 40 centimetrov, táto hodnota bola u pôvodného senzoru 80 centimetrov. Pri využívaní blízkeho režimu môžeme stáť bezprostredne pred Kinectom, senzor je použiteľný aj v menších miestnostiach (pri hraní hier s Kinectom na konzole Xbox, je doporučená vzdialenosť medzi senzorom a hráčom 1,8 m, v prípade dvoch hráčov 2,5 m). Vzdialenosti, pri ktorých je senzor použiteľný, sú znázornené na Obrázku 13. V normálnom režime Kinect rozpozná objekty od vzdialenosti 80 centimetrov do štyroch metrov, v blízkom režime sa tieto hodnoty zmenia na 40 centimetrov a tri metre.



Obrázok 13 Rozdiel medzi normálnym a blízkym režimom senzoru Kinect for Windows[18]

Na konci marca roku 2012 sa na oficiálnom blogu projektu Kinect for Windows objavili informácie o *verzii 1.5* vývojového nástroja, ktorá by mala byť dostupná na konci mája. Nová verzia bude obsahovať softvér *Kinect Studio*, ktorý bude veľkou pomocou vývojárov pri ladení aplikácií. S touto novou aplikáciou môžu vývojári nahrávať a prehrávať videá a ich pomocou ladiť svoje aplikácie. Okrem toho nový nástroj podporuje rozpoznávanie hornej časti tela človeka. Táto funkcia umožňuje sledovanie sediaceho užívateľa v normálnom a blízkom režime – Kinect môžeme umiestniť vedľa monitoru, a aj tak bude schopný rozpoznať a sledovať sediaceho užívateľa. Rozhranie rozpoznávania reči sa rozširuje o ďalšie štyri jazyky: francúzština, španielčina, taliančina a japončina.[19]

Ako už to bolo zmienené, beta verzia Kinect SDK sa mohla využívať len na nekomerčné účely. Aplikácie vytvorené výslednou verziou vývojového nástroja Kinect for Windows SDK a používané spolu s Kinectom pre Windows môžu byť použiteľné už aj v komerčných projektoch. Zavedenie takýchto aplikácií priamo u zákazníkov je možné pomocou stiahnuteľného inštalátoru s runtime komponentmi Kinect for Windows a s ovládačom pre Kinect. Vývojový nástroj Kinect for Windows SDK, vývojové prostredie Visual Studio 2010 Express, runtime knižnice a ovládač senzoru sú bezplatne dostupné a stiahnuteľné z webových stránok Microsoftu, kúpiť sa musí len vlastný senzor. Bohužiaľ Kinect pre Windows, verzia senzoru, ktorá by podľa Microsoftu mala byť použitá s PC, je

[18] Zdroj obrázku: Near Mode: What it is (and isn't) (http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/20/near-mode-what-it-is-and-isn-t.aspx).
[19] What's Ahead: A Sneak Peek (http://blogs.msdn.com/b/kinectforwindows/archive/2012/03/26/what-s-ahead-a-sneak-peek.aspx).

dostupný len v jedenástich krajinách, medzi ktorými nie je ani Česká republika ani Slovensko. Cena senzoru pre Windows je približne o šesťdesiat percent vyššia oproti verzii senzoru pre Xbox, a sľúbené nižšie akademické ceny platia len vo Spojených Štátoch.

Vývojový nástroj Kinect for Windows SDK je vďaka totožnému hardvéru plne kompatibilný so staršou verziou senzoru pre Xbox. Pri použití Kinectu pre Xbox narazíme na isté obmedzenia: Kinect pre Xbox nie je možné použiť v komerčných projektoch a nie je kompatibilný s runtime komponentmi.

V implementačnej časti tejto práce zaoberajúcej sa vývojom prirodzeného užívateľského rozhrania som použil ovládače ku Kinectu a vývojové nástroje z Kinect for Windows SDK. Kvôli problémom s dostupnosťou Kinectu pre Windows a jeho vyššej ceny som použil Xbox-ovú verziu senzoru.

3.4.1 Vývoj aplikácií v Kinect for Windows SDK

V nasledujúcej podkapitole predstavím použité princípy a dôležitejšie triedy vývojového nástroju Kinect for Windows SDK na základe (7).

Vývojový nástroj podporuje jazyky *C++*, *C#* a *Visual Basic* použitím vývojového prostredia *Microsoft Visual Studio 2010*. Oficiálne hardvérové a softvérové požiadavky SDK sú v Prílohe 1.

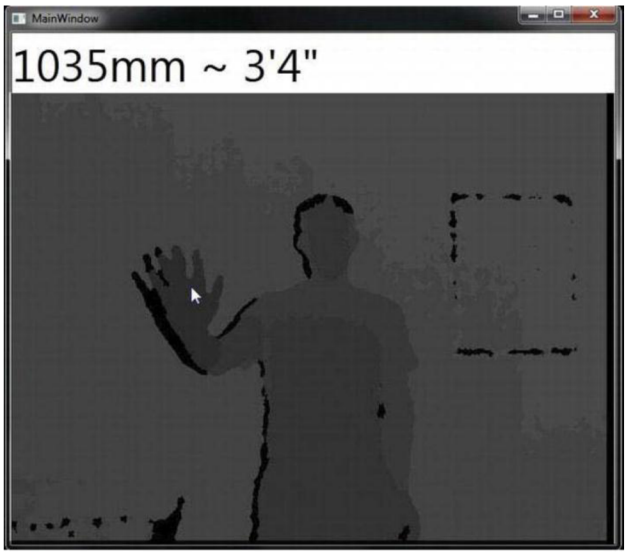
Pri vývoji aplikácií v Kinect SDK pracujeme s tromi dátovými prúdmi z Kinectu:

- Farebný obraz z RGB kamery,
- Hĺbkový obraz generovaný pomocou infračerveného senzoru,
- Dátový prúd, ktorý obsahuje informácie o kostre sledovaných užívateľov,

Získať aktuálne snímky/dáta z týchto prúdov môžeme dvomi spôsobmi: využívaním udalostí alebo manuálnou žiadosťou o dáta (*polling*). V prvom prípade dostane naša aplikácia udalosť od dátového prúdu o tom, že nasledujúca snímka je pripravená. Manuálny polling je výhodný v prípadoch, keď naša aplikácia spracováva snímky z Kinectu v aplikačnom cykle (napr. v prípade hier – herný cyklus). Aktuálnu snímku z prúdu môžeme získať v každom cyklu našej aplikácie.

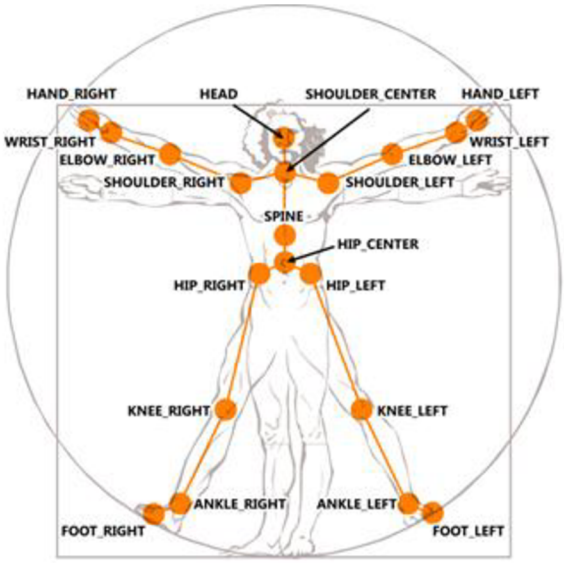
Hĺbkový obraz z Kinectu má rozlíšenie 640x480, každý pixel reprezentuje hodnoty vzdialenosti od senzoru v šestnástich bitoch, z ktorých prvé tri bity reprezentujú index užívateľa. Pomocou hodnoty indexu môžeme zistiť, ktoré pixely obrazu patria k postave užívateľa, a ktoré patria k pozadiu. Využívaním týchto hodnôt sme schopní jednoduchým spôsobom vytvoriť rozšírenú realitu. Mapovaním pixelov užívateľa z hĺbkového obrazu na farebný obraz RGB kamery získame farebné pixely reprezentujúce sledovaného človeka - segmentovali sme postavu užívateľa z farebného obrazu. Premietaním farebných pixelov postavy na virtuálne pozadie získame rozšírenú realitu.

Na obrázku 14 vidíme hĺbkový obraz zo senzoru. Rozdiely vo vzdialenosti sú znázornené rôznymi odtieňmi šedej farby.



Obrázok 14 Hĺbkový obráz z Kinectu, na hornej lište je vzdialenosť objektu od senzora (7)

Tretí dátový prúd Kinectu obsahuje informáciu o virtuálnej **kostre sledovaných užívateľov** (*skeleton tracking*). Z tohto prúdu dostaneme informáciu o polohe jednotlivých častí užívateľovho tela. Virtuálna kostra (ďalej len kostra) sledovaná senzorom pozostáva z 20 bodov, ktoré reprezentujú kĺby a ďalšie časti ľudského tela – SDK definuje tieto body ako virtuálne kĺby (*joints*, ďalej len kĺby).



Obrázok 15 Kostra užívateľa v Kinect for Windows SDK: kostra pozostáva z 20 kĺbov[20]

Pohyb sledovaných kĺbov môže byť v niektorých prípadoch nesúvislý. Príčinou tohto javu je pravdepodobne buď výkon našej aplikácie, slabý výkon Kinectu alebo chovanie užívateľa. SDK obsahuje funkciu pre korekciu a vyhľadenie takýchto anomálií. Korekčná funkcia upravuje menšie vibrácie a trasenie sa kĺbov. Vyhľadávanie funguje na základe algoritmu *Holt double exponential smoothing*[21].

SDK poskytuje funkciu na výber kostry užívateľa, pre prípad že pred Kinectom stojí viac ľudí. Napríklad: keď pred Kinectom stojí šesť ľudí, senzor rozpozná každý z nich a je schopný sledovať kostru dvoch z šiestich rozpoznaných užívateľov. Dáva nám možnosť vybrať, z ktorej z týchto dvoch

[20] Zdroj obrázku: http://msdn.microsoft.com/en-us/library/hh438998.aspx.
[21] Double Exponential smoothing (http://en.wikipedia.org/wiki/Exponential_smoothing#Double_exponential_smoothing).

má čerpať informácie. V predvolenom nastavení vykoná túto úlohu rozpoznávací algoritmus, ktorého voľba je v podstate nepredvídateľná. Preto nám daná funkcia umožňuje možnosť voľby konkrétnej kostry, s ktorou chceme pracovať.

Rozpoznávanie reči v Kinect for Windows SDK: Pomocou štyroch mikrofónov Kinectu a Kinect for Windows SDK môžeme rozpoznať hlasové povely a reč človeka, a zistiť smer prichádzajúceho hlasu (lokalizácia užívateľa podľa hlasu). SDK obsahuje implementované nástroje, ktoré pomáhajú pri vyžití týchto funkcií: zrušenie akustických ozvien (*AEC*[22]), potlačenie šumu (*noise suppression*), *automatic gain control* (*AGC*[23]).

[22] Acoustic echo cancellation (http://en.wikipedia.org/wiki/Echo_cancellation#Acoustic_echo_cancellation).
[23] Automatic gain control (http://en.wikipedia.org/wiki/Automatic_gain_control).

4 Prirodzené užívateľské rozhranie a Kinect

V predchádzajúcich kapitolách som sa zaoberal s prirodzenými užívateľskými rozhraniami a so senzorom Kinect. Tento typ užívateľského rozhrania používa rôzne prejavy užívateľa na ovládanie počítača. Kinect je zariadenie, ktorým je možné tieto užívateľské prejavy sledovať. Senzor disponuje s farebnou a hĺbkovou kamerou na rozpoznávanie a sledovanie užívateľa a so štyrmi mikrofónmi, ktoré umožnia rozpoznávať ľudskú reč.

V tejto kapitole skombinujem vedomosti z predchádzajúcich kapitol o prirodzených rozhraniach a o Kinectu a popíšem spojenie rozhrania s hardvérom senzoru. Toto spojenie je vlastne predmetom praktickej časti mojej diplomovej práce:

Cieľom projektu je návrh takého spôsobu ovládania klasických desktopových aplikácií pomocou senzoru Kinect, ktorý je efektívny a z hľadiska užívateľa prívetivý. Implementácia výsledného návrhu by mala predstaviť rozhranie na emuláciu klasických vstupných periférií ako myš a klávesnica.

V tejto kapitole sa venujem návrhu spôsobov ovládania desktopových aplikácií a integrácií týchto spôsobov do prirodzeného užívateľského rozhrania. Nasledujúca kapitola sa ďalej zaoberá implementáciou prirodzeného rozhrania s navrhovanými spôsobmi ovládania aplikácií.

4.1 Spôsoby ovládania klasických desktopových aplikácií pomocou Kinectu

4.1.1 Ovládanie klasických desktopových aplikácií

Pred návrhom ovládania desktopových aplikácií pomocou Kinectu, musíme preskúmať, ako sa tieto aplikácie ovládajú inými vstupnými zariadeniami. Na ovládanie takýchto aplikácií v dnešnej dobe používame klávesnicu a myš. Fungovanie zariadenia, ako sú touchpad, trackpoint a špeciálne herné ovládače je založené na rovnakých princípoch ako hore spomenuté dve periférie. Kinect obsahuje kameru, hĺbkový senzor a mikrofóny, preto sa týmito vstupnými zariadeniami nie je potrebné zvlášť zaoberať, sú popísané v rámci popisu senzoru Kinect. Dotykové obrazovky, dátové rukavice a haptické senzory v dnešnej dobe ešte nie sú bežne používanými zariadeniami pri ovládaní desktopových aplikácií. Pri návrhu možných spôsobov ovládania aplikácií pomocou Kinectu sa teda musíme sústrediť len na metódy vstupu pomocou klávesnice a myši. Skúmanie princípov týchto vstupných zariadení ale neznamená, že ovládanie pomocou Kinectu musí tieto princípy kopírovať.

Ovládanie aplikácií pomocou myši zahrňuje funkcie ako je pohyb s kurzorom, výber objektu/objektov, premiestenie objektu/objektov, zobrazenie ponuky, posúvanie okien alebo rolovanie ich obsahu. Existujú aplikácie, ktoré k vyššie zmieneným funkciám pridali aj vlastné funkcie, ktoré sú ale závislé na konkrétnej skúmanej aplikácii (napr. u počítačových hier výstrel, ovládanie postavy).

Na písanie znakov do textového pole slúži najčastejšie klávesnica. Táto činnosť zahrňuje okrem písania znakov aj pomocné funkcie ako medzera, nový riadok, vymazanie, pohyb textového kurzoru, výber písania veľkých písmen, atď. Tieto funkcie majú väčšinou vlastný kláves na klávesnici. Pri skúmaní ovládania aplikácií pomocou klávesnice sa musíme zmieniť aj o klávesových skratkách. Pri stlačením istých klávesov naraz môžeme spustiť rôzne funkcie aplikácií.

Desktopové aplikácie obsahujú aj funkcie, ktoré nie sú priamo viazané na tieto dve vstupné zariadenia. Myslím tým hlavne funkcie operačných systémov ako zavrieť, minimalizovať a maximalizovať okno, striedať okná na popredí, zobraziť plochu systému, striedať medzi plochami systému. Tieto funkcie systémov sú pri používaní klávesnice a myši spustiteľné rôznymi spôsobmi.

V predchádzajúcich odsekoch sme preskúmali ovládanie desktopových aplikácií z pohľadu vstupného zariadenia. Aby sme mohli navrhnúť ovládacie metódy takýchto aplikácií využívajúcich Kinect, musíme sa zaoberať aj s ovládacími prvkami používanými v desktopových systémoch.

4.1.2 Grafické užívateľské rozhranie desktopových aplikácií

Desktopové aplikácie sú súčasťou grafického užívateľského rozhrania dnešných operačných systémov. Aby sme pochopili spôsob ovládania komponentov aplikácie, musíme najprv preskúmať tieto užívateľské rozhrania.

Grafické užívateľské rozhrania dnešných desktopových operačných systémov používajú koncept *WIMP* (*Windows*, *Icons*, *Menus*, *Pointer*). Hlavnými prvkami týchto systémov sú okná, ikony, ponuky a ukazovateľ. V nasledujúcich odsekoch sú popísané tieto základné prvky na základe [30]:

- **Okná** sú základnými zobrazovacími prvkami, grafické rozhranie každej aplikácie sa zobrazuje v jednom alebo vo viacerých oknách.
- **Ikony** reprezentujú rôzne typy súborov, ako sú napr. dokumenty, spustiteľné programy alebo zložky.
- **Ponuky** obsahujú rôzne príkazy zoskupené do zoznamov, a môžu tvoriť aj komplexnejšie štruktúry. Užívateľ si nemusí zapamätať príkazy, môže si ich vyhľadať pomocou ponuky.
- **Ukazovateľ** reprezentuje vstupné informácie od užívateľa. Pomocou takýchto ukazovateľov sa desktopové systémy vlastne ovládajú. Systémy používajú dve typy ukazovateľov: textový kurzor (*caret*) a kurzor myši (*pointer*). Textový kurzor ukazuje miesto, kam budú nasledujúce znaky písané. Reprezentuje textový vstup od užívateľa. Kurzor myši je hlavným prostriedkom interakcie s ostanými základnými prvkami WIMP systému, ako sú okná, ikony a ponuky.

Aplikácie môžu zobraziť svoje grafické užívateľské rozhranie pomocou okien. Okná pozostávajú z tzv. *ovládacích prvkov* alebo *widgetov*[24]. Tieto prvky tvoria základné rozhranie aplikácie pre komunikáciu s užívateľom: tlačidlo, textové pole, rôzne typy ponuky.

Ovládanie desktopových aplikácií je možné využívaním textového kurzoru a kurzoru myši. Pomocou týchto ukazovateľov je schopný užívateľ pracovať s widgetmi aplikáciami.

[24] Graphical user interface elements (http://en.wikipedia.org/wiki/Graphical_user_interface_elements).

4.1.3 Ovládanie desktopových aplikácií pohybmi užívateľa

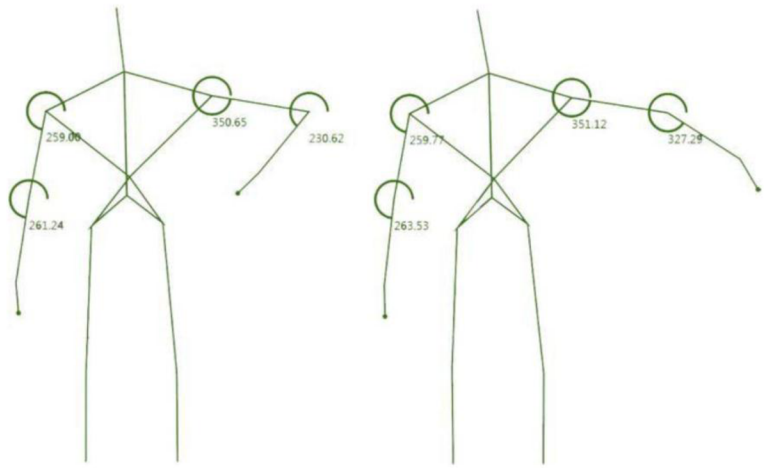
O ovládaní počítača pomocou pohybov užívateľa som už písal v kapitole o prirodzených užívateľských rozhraniach. Tieto rozhrania používajú statické a dynamické prejavy človeka na spustenie rôznych funkcií operačných systémov. Grafické užívateľské rozhrania operačných systémov a desktopových aplikácií, ktoré na nich bežia sú veľmi podobné. Ich štruktúra a komponenty sú totožné: v oknách operačného systému a aplikácií sa používajú tie isté ovládacie prvky, widgety na dosiahnutie totožnej funkčnosti (napr.: ovládacie prvky operačných systémov Windows sú totožné, ako prvky používané v aplikáciách *Windows Forms* alebo *WPF*[25]). Z toho plynie, že statickými a dynamickými prejavmi prirodzených užívateľských rozhraní, pomocou ktorých môžeme počítač ovládať, môžeme ovládať aj desktopové aplikácie. Tieto prejavy sú statické pózy a dynamické gestá.

Hĺbkový senzor Kinectu nám umožňuje rozpoznávanie postavy užívateľa a sledovanie jeho pohybov. Zo senzoru dostaneme tri dátové prúdy, farebný obraz, hĺbkový obraz a polohu kostry užívateľa. Z týchto troch prúdov sú pre nás najdôležitejšie posledné dva: algoritmy pre rozpoznávanie užívateľa pred senzorom sú založené na analýze hĺbkových snímok. Sledovať polohu užívateľa a vzájomnú pozíciu častí jeho tela môžeme pomocou tretieho prúdu, ktorý obsahuje informácie o virtuálnej kostre užívateľa. Táto kostra pozostáva z dvadsiatich virtuálnych kĺbov, ktoré reprezentujú nasledujúce časti tela: ľavá a pravá dlaň, zápästie, lakeť, rameno, noha, členok, koleno, bok, chrbtica a hlava. Pomocou sledovania polohy týchto kĺbov v skúmanom priestoru, môžeme sledovať pohyby užívateľa.

Kombináciou statických póz a dynamických gest prirodzených užívateľských rozhraní a schopnosti Kinectu sledovať pohyby užívateľa sme schopní ovládať desktopové aplikácie. Presnejšie, pre ovládanie aplikácií môžeme použiť pózy a gestá vykonávané sledovanou virtuálnou kostrou užívateľa:

- **Statické pózy a Kinect**: Statická póza je taký spôsob držania tela, pri ktorom sa užívateľ nehýbe. Pózu sú určené vzájomnou pozíciou častí užívateľovho tela, napr. spôsob držania ruky, naklonenie nôh, uhol naklonenia hlavy. Od senzoru Kinect dostaneme informácie o týchto vzájomných polohách časti tela len vo forme polohy dvadsiatich virtuálnych kĺbov virtuálnej kostry. Presnejšie, poloha kĺbov je daná súradnicami v karteziánskej sústave súradníc Kinectu (začiatok sústavy je senzor Kinect). Rozpoznania statickej pózy môžeme dosiahnuť porovnaním polohy týchto kĺbov, alebo vypočítaním uhlov medzi *kosťami* kostry (kosťami kostry sa rozumie spojovacia priamka medzi dvomi kĺbmi). Na Obrázku 16 je znázornená virtuálna kostra a hodnoty vypočítaných uhlov medzi kosťami a kĺbmi.

[25] Windows Presentation Foundation: <msdn.microsoft.com/en-us/library/ms754130.aspx>.

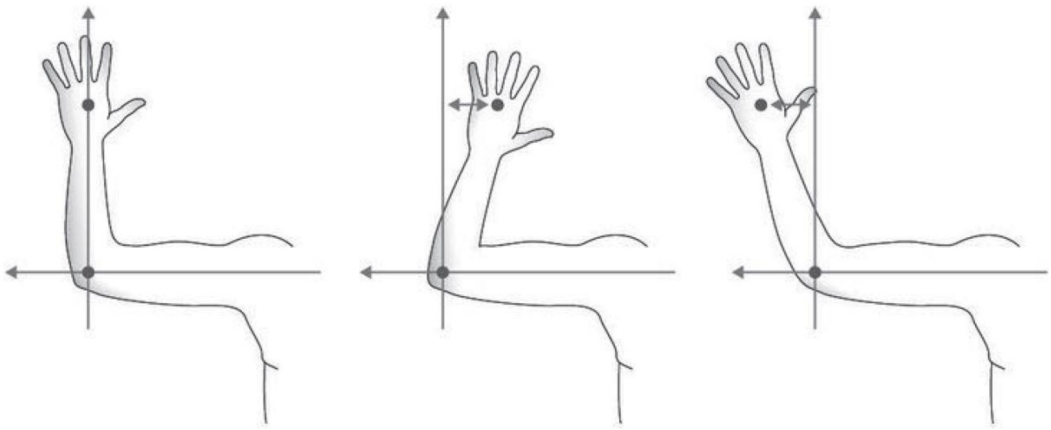


Obrázok 16 Virtuálna kostra užívateľa, a uhly medzi kĺbmi a kosťami virtuálnej kostry užívateľa (7)

- **Dynamické gestá a Kinect**: Dynamické gesto je taký pohyb tela, ktorý prenáša informácie. Tieto informácie môžeme detekovať pomocou senzoru Kinect, a následne ich využitím môžeme ovládať desktopové aplikácie. V definícii gesta, ktorá sa nachádza v kapitole 2.3.3 nie je presne špecifikované, ktorými časťami tela môže užívateľ gestá vykonávať, a ktoré pohyby vlastne môžeme nazvať gestom. Preto pri sledovaní pohybov užívateľa je nutné pozorovať jeho celé telo, gesto totiž môže byť vyjadrené pohybom nohy, mávnutím ruky alebo aj naklonením hlavy.

  Pre sledovanie pohybov užívateľa môžeme použiť len informácie, ktoré sme používal aj na rozpoznávanie statických póz, t.j. polohu virtuálnej kostry užívateľa. Pomocou týchto informácií sme schopní presne určiť pozíciu skúmanej časti tela. Gesto je ale dynamický prejav, rozpoznať ho z aktuálnej snímky virtuálnej kostry je nemožné. Metóda, ktorú sme používali u statických póz u gest teda nefunguje. Aby sme mohli detekovať gestá, potrebujeme informáciu o predchádzajúcej polohe skúmanej časti tela. Je nutné nejakým spôsobom uložiť informáciu o aktuálnej polohe pohybujúceho sa tela. To znamená, že pri sledovaní virtuálnej kostry užívateľa musíme uložiť aktuálne súradnice skúmaných alebo všetkých kĺbov. Takto budeme mať v pamäti uložené súradnice všetkých bodov, ktoré boli zahrnuté do pohybu pri vykonávaní gesta.

  Pre rozpoznávanie dynamických gest môžeme použiť aj statické pózy. Miesto toho, aby sme počas spracovania aktuálnej snímky ukladali súradnice skúmaných kĺbov, môžeme uložiť informácie o aktuálnej póze užívateľa. Póza obsahuje informácie o držaní tela, preto medzi nimi nájdeme aj aktuálnu polohu skúmanej časti tela.



Obrázok 17 Poloha zápästia voči lakte počas predvedenia gesta mávanie rukou (7)

Pri pozorovaní pohybov užívateľa môžeme desktopové aplikácie ovládať pomocou statických póz a dynamických gest, pričom senzor Kinect slúži na rozpoznanie týchto prejavov.

4.1.4 Ovládanie desktopových aplikácií ľudskou rečou

Hlasová komunikácia medzi ľuďmi je najprirodzenejším spôsobom medziľudskej komunikácie. Napriek tomu, že vývoj v oboru rozpoznávania reči je veľmi pokročilý, ovládanie počítača a desktopových aplikácií je stále nepredstaviteľné. Kvôli rôznorodosti ľudskej reči a rôznym jazykom, počítač ešte nie je schopný rozpoznať každé slovo a každú vyslovenú vetu.

Grafické užívateľské rozhrania dnešných desktopových aplikácií sú založené na ovládaní pomocou klávesnice a myši. Emulovať funkčnosť týchto vstupných zariadení pomocou rozpoznávania reči je veľmi ťažké. Keď skúmame využívanie reči pri ovládaní grafických rozhraní z pohľadu aplikácie, je jasné, že ovládať všetky widgety a komponenty hlasom je veľmi obtiažne (napr.: použitie reči na ovládanie komponentov *check box*, *radio button*, *context menu*, *combo box*, atď). Aby sme boli schopní ovládať tieto prvky jedine pomocou hlasových rozkazov, museli by sme si pamätať názov každého komponentu.

Spôsoby ovládania desktopových aplikácií pomocou reči môžeme rozčleniť do dvoch skupín: ovládanie hlasovými povelmi a ovládanie plynulou rečou.

Hlasové povely môžeme chápať ako príkazy príkazového riadku bez argumentov. Každý hlasový príkaz by predstavoval jednu funkciu aplikácie, a pri jeho vyslovení by sa táto funkcia spustila. Kvôli tomu, že počet takýchto povelov je obmedzený, tento spôsob ovládania by bol použiteľný len v prípadoch aplikácií s jednoduchým grafickým rozhraním. Príkladom tohto spôsobu ovládania aplikácie by mohlo byť ovládanie známej hry *sokoban* pomocou štyroch hlasových povelov: hore, dolu, doľava, doprava.

Použiteľnejším a zaujímavejším spôsobom ovládania desktopových aplikácií by bolo využitie **rozpoznávania plynulej reči**. Myšlienka, podľa ktorej by sme boli schopní ovládať počítač a aplikácie len rozprávaním, je fascinujúca.

Emulovať klávesnicu pri písaní rozpoznávaním reči by bola veľmi jednoduchá. Miesto toho, aby sme text písali klasicky, t.j. stlačením klávesov, systém by naše slová rozpoznal a napísal. Emulácia myši by už nebola taká jednoduchá, pretože polohovanie kurzoru potrebuje plynulý pohyb od užívateľa. Vysloviť slovo „doľava“ viackrát za sebou, kým sa nás kurzor nedostane nad hľadaný objekt, je veľmi neefektívne a pre užívateľa nepohodlné. Lepším spôsobom by bolo použitie názvov komponentov, s ktorými chceme pracovať. Vďaka tomu, že systém rozpozná nielen hlasové povely,

ale aj plynulú reč v prípade, že nepoznáme presný názov daného komponentu, môžeme jeho polohu voči ostaným komponentom popísať slovami. Takýto spôsob ovládania desktopových aplikácií by z pohľadu užívateľa fungoval veľmi jednoducho: „pri ovládaní aplikácie len musím vysloviť čo od nej chcem, akú funkciu potrebujem spustiť“

Využitím vyššie popísaných princípov ovládania a princípu projektu *Put That There* (8) môžeme navrhnúť systém na ovládanie desktopových aplikácií. V tejto práci sa okrem hlasových príkazov používa aj polohovacie zariadenie vo forme kurzora. V mojom návrhe nahradím toto zariadenie vyslovením požadovanej polohy kurzoru. Vyslovenie slova „there“ ale okrem určenia polohy slúži aj ako posledné slovo príkazu: po jeho vyslovení aplikácia spustí vykonávanie požiadavky. V našom systému bude takýmto koncovým slovom slovo „teraz“, čím prikážeme systému, aby náš príkaz vykonal ihneď. Ovládanie aplikácií pomocou tohto systému by mohlo znieť nasledovne:

„Stlač tlačidlo – OK -, teraz.“

„Píš text – Ahoj svet -, teraz.“

„Stlač tlačidlo vpravo od ikony – Nový dokument-, teraz.“

„Minimalizuj aktuálnu aplikáciu, teraz.“

„Zavri aplikáciu Notepad, teraz.“

4.2 Návrh prirodzeného užívateľského rozhrania pracujúceho s Kinectom

V predchádzajúcej podkapitole som predstavil štyri spôsoby, ako by sme mohli ovládať desktopové aplikácie pomocou senzoru Kinect. Využitím dátového prúdu hĺbkového obrazu a virtuálnej kostry je možné sledovať pohyb užívateľa. Navrhované spôsoby ovládania sú založené na detekcii statických póz a dynamických gest vykonávaných užívateľom. Ďalšie dve navrhované spôsoby ovládania využívajú ľudskú reč, rozpoznaním jednotlivých slov reči určia príkazy užívateľa. Aby som tieto navrhované spôsoby mohol v ďalších častiach tejto práce efektívne použiť, musím ich integrovať do prirodzeného užívateľského rozhrania.

Výsledné navrhované rozhranie bude používať na ovládanie aplikácií statické pózy, dynamické gestá a hlasové povely užívateľa. Pred vlastným návrhom pripomeniem ešte požiadavky na prirodzené užívateľské rozhranie plynúce z definície (kapitola 2.2).

Prirodzené užívateľské rozhrania využívajú existujúce schopnosti užívateľa, a interakcia s obsahom pomocou rozhrania je priama.

- **Rozhranie využíva existujúce schopnosti**: Ovládanie rozhrania musí byť intuitívne pre užívateľa, pri ovládaní môže využiť svoje existujúce schopnosti. Pre navrhované rozhranie to znamená, že pózy a gestá musia byť jednoduché a pochopiteľné. Hlasové povely splňujú túto požiadavku, vyslovené príkazy a reč je časťou každodennej komunikácie.
- **Priama interakcia s obsahom**: Rozpoznávacie a sledovacie schopnosti senzoru Kinect zaistia splnenie požiadavky. Z pohľadu užívateľa je interakcia s prvkami direktná, nie je potrebné použiť pomocné vstupné zariadenie.

Okrem splnení týchto požiadaviek musí byť rozhranie efektívne a prívetivé pre užívateľa. Efektívnosti týchto spôsobov ovládania aplikácií dosiahneme ich integráciou do prirodzeného užívateľského rozhrania. Táto integrácia umožní použitie týchto spôsobov ovládania paralelne. Dosiahnuť užívateľskej prívetivosti je už ťažšia úloha, ktorá vyžaduje spätnú väzbu od užívateľov počas návrhu a implementácie rozhrania. Aj dokonale navrhnuté a implementované rozhranie môže byť pre koncového užívateľa nepoužiteľné, ak použité riešenia neboli testované. Je dôležité zmieniť sa aj o tom, že rozhranie, ktoré je prívetivé pre programátora, nemusí byť prívetivé aj pre koncového užívateľa.

Implementácia rozhrania by mala emulovať klasické vstupné zariadenia ako myš a klávesnica. Aplikácie by mali byť ovládateľné len pomocou navrhovaného rozhrania a Kinectu, bez ďalších vstupných periférií. V desktopových operačných systémoch textový kurzor a kurzor myši je viazaný na užívateľské vstupy: Tieto dva ukazovatele sú úzko spojené s funkčnosťou klávesnici a myši.

Navrhované spôsoby ovládania kurzoru myši a ostaných funkcií sú založené na sledovaní virtuálnej kostry užívateľa. Funkcie sú ovládané pózami a gestami vykonané virtuálnymi kĺbmi.

Návrh prirodzeného rozhrania som rozdelil na tri časti: ovládanie kurzoru myši, desktopu a ovládanie textového kurzoru.

4.2.1 Ovládanie kurzoru myši

Pohyb kurzorom myši je ovládaný pravou dlaňou. Pohyb dlane po vodorovnej a zvislej ose priamo zmení pozíciu kurzora – poloha kurzora je spojená s polohou dlane užívateľa. Aby sa kurzor

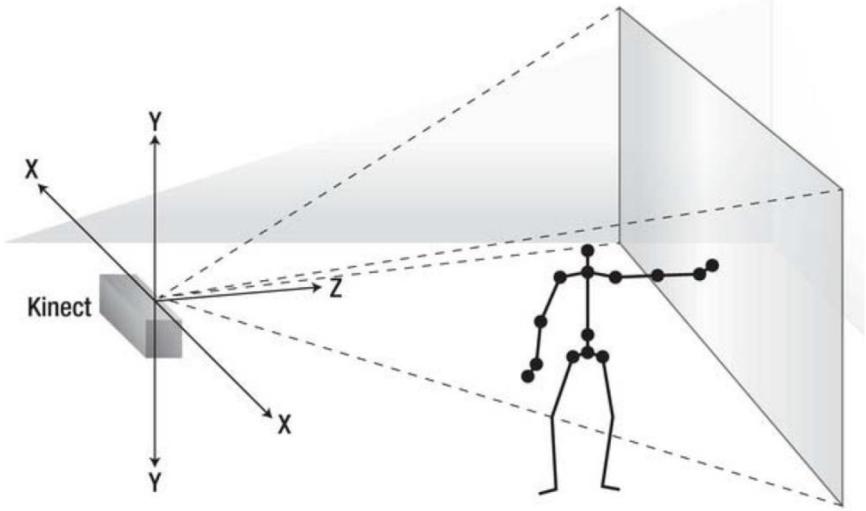
pohyboval, musí užívateľ hýbať rukou v imaginárnom štvorci vo vzduchu. Tento štvorec predstavuje obrazovku a je umiestnený po pravej strane užívateľa. Keď sa užívateľova ruka dostane von z tohto štvorca, kurzor sa zastaví na hranici obrazovky. Kurzorom myši môžeme pohybovať len v tom prípade, ak je dlaň ľavej ruky pod virtuálnym kĺbom *spine*, ktorý sa nachádza v oblasti pupku.

Pravé a ľavé kliknutie myšou: Pre kliknutie je potrebné aby bola pravá ruka nad kĺbom spine a ukazovala na miesto kam chceme kliknúť. Kliknutie samotné ovládame ľavou rukou, ktorá musí byť pod týmto kĺbom. Pohyb ľavej ruky doľava znamená kliknutie ľavým tlačidlom, pohyb doprava znamená kliknutie pravým tlačidlom.

Na ovládanie tretieho, **prostredného tlačidla** sa používa pravá ruka. Doteraz som používal len pohyby po osách *x* a *y*. Na ovládanie tohto tlačidla sa ruka pohybuje aj po ose *z*: pohybom pravej dlane dopredu stlačím tlačidlo a ďalej už môžem hýbať kurzorom podľa vyššie popísaného postupu. Tlačidlo uvoľním pohybom dlane dozadu.

Odôvodnenie návrhu:

- Dôvod, prečo som pre pravé a ľavé kliknutie myšou nepoužíval pohyb pravej ruky dopredu, sa skrýva v štruktúre zorného pola senzoru Kinect, ktoré je zobrazené na Obrázku 18. Pri pohybu ruky dopredu sa totiž nemení len súradnica *z* sledovaného kĺbu, ale menia sa aj súradnice *x* a *y*. Preto kurzor pri pohybu pravej ruky dopredu nezostáva na mieste, ale hýbe sa. Tento pohyb som použil na ovládanie prostredného tlačidla myši, pretože tlačidlo sa väčšinou používa na rolovanie, pričom malé pohyby kurzoru nevadia. Rolovanie teda môže fungovať aj keď kurzor nezostáva na mieste.



Obrázok 18 Zorné pole senzoru Kinect (7)

- Poloha rúk oproti kĺbu spine (t.j. ľavá ruka pod ním, pravá nad ním) je dôležitá, pretože ich iná poloha (napr. pravá pod ním a ľavá nad) znamená ovládanie iných funkcií rozhrania.

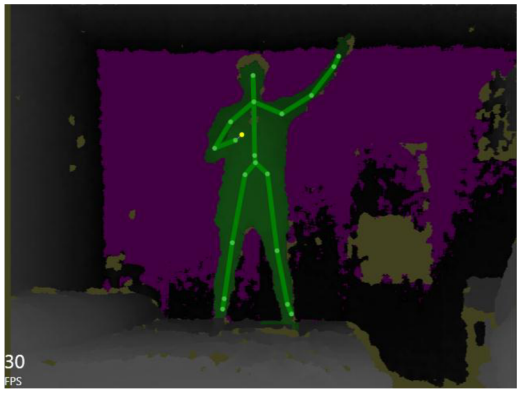
4.2.2 Ovládanie desktopových funkcií

Polohy a pohyby pravej ruky ovládajú funkcie: zavrieť okno, ukáž plochu, minimalizovať a maximalizovať okno, zmeň okno v popredí (*switch top-level window*). Tieto funkcie sa ovládajú len pohybmi pravej ruky, ľavá dlaň je pritom vždy medzi kĺbom spine a ľavým ramenom. Medzi dvoma pózami je vždy nutné ruku uvoľniť do základnej polohy – pripažiť.

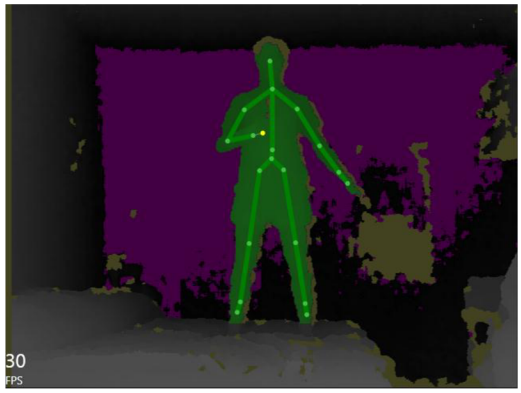
- **Okno sa zatvára** pomocou pózy pri ktorej je pravá ruka vystretá a s telom uzatvára uhol 135° (obrázok 19).
- Keď je pravá ruka stále vystretá, ale s telom uzatvára uhol 45°, na obrazovke sa objaví **plocha** (obrázok 20).

Funkcie **maximalizovať** a **minimalizovať** sa ovládajú pomocou gesta *swipe*[26] vo zvislom smere. Pohybom ruky zdola nahor okno maximalizujeme, opačným pohybom ruky (zhora nadol) ho minimalizujeme.

- **Zmeniť okno v popredí** môžeme taktiež pomocou gesta swipe, tentoraz ho použijeme vo vodorovnom smere.



Obrázok 19 Zatvorenie okna



Obrázok 20 Ukázať plochu

Polohy a pohyby ľavej ruky predstavujú funkcie nasledujúcich klávesov na klávesnici: Esc, Enter, smerové šípky a kláves Štart. Ku každému klávesu patrí iná póza alebo gesto, pri ktorých sa ľavá ruka voľne pohybuje, pravá ruka je v pripažení a pravá dlaň je pod kĺbom spine. Rovnako ako u pravej ruky, aj v prípade ľavej je nutné medzi jednotlivými pózami vždy pripažiť.

- **Kláves Esc** stlačíme nasledujúcou pózou: pravá ruka v pripažení; ľavá ruka vystretá, uhol medzi ľavou rukou a telom je 135 stupňov (obrázok 21).
- Na stlačenie **klávesy Štart** sa používa póza pri ktorej vystretá ľavá ruka užívateľa uzatvára uhol 45° s jeho telom, jeho pravá ruka je v pripažení (obrázok 22).
- Na stlačenie **klávesy Enter** sa používa podobný pohyb ako na ovládanie prostredného tlačidla myši: pohyb dlane dopredu.
- **Šípky** sa ovládajú gestom swipe po horizontálnej a vertikálnej ose vo smeru požadovanej šípky.

[26] Potiahnutie rukou

Obrázok 21 Kláves Esc

Obrázok 22 Ponuka Štart

Odôvodnenie návrhu:

- K funkciám, ktoré sa ovládajú pózami som priradil polohu ruky vždy podľa toho, kde sa tieto funkcie na obrazovke najčastejšie vyskytujú: ikona na zatvorenie okna je spravidla vpravo hore, kým funkcia ukáž plochu je v pravom dolnom rohu (Windows 7). Rovnako som postupoval v prípade klávesy Esc a Štart, pózami na ich ovládanie vyjadrujem ich umiestnenie na klávesnici (vľavo hore a dole).
- Pohybmi dlane po vodorovnej a zvislej ose sa šípky ovládajú preto, lebo tieto pohyby sú pre užívateľa logické a jednoznačné. Pre rovnaký dôvod som na ovládanie funkcií minimalizovať a maximalizovať zvolil gesto swipe vo zvislom smere.
- Pri návrhu gesta pre ovládanie funkcie zmeniť okno v popredí som sa inšpiroval softvérmi chytrých telefónov, ktoré používajú tento pohyb na listovanie medzi obrázkami.

4.2.3 Ovládanie textového kurzoru

Ovládanie textového kurzoru v navrhovanom prirodzenom rozhraní je možné pomocou dynamických gest. Užívateľ komunikuje s rozhraním len pomocou svojich pohybov. Pri vykonaní ovládacích gest potrebuje spätnú väzbu o aktuálnej polohe svojej ruky. Preto navrhované rozhranie má aj grafické užívateľské rozhranie, na ktorom môže užívateľ vidieť polohu svojej ruky a aktuálny stav vykonávaného gesta.

Ovládanie textového kurzoru znamená emuláciu funkčnosti klávesnice. Navrhované rozhranie nahradí toto vstupné zariadenie virtuálnou klávesnicou, ktorá je schopná emulovať všetky funkcie periférie. K ovládaniu tejto virtuálnej klávesnice používame obe ruky. Ľavou rukou určujeme, aké znaky chceme konkrétne písať: malé alebo veľké písmená, číslice či interpunkciu. Na napísanie jednotlivých znakov sa následne používa pravá ruka.

Na virtuálnej klávesnici sú virtuálne klávesy, pomocou ktorých sa môže napísať text. Jednotlivé znaky textu môže užívateľ napísať stlačením virtuálnych klávesov klávesnice v správnom poradí. Najprv potrebuje dotykom vybrať požadované písmeno alebo iný znak, a následne ho potrebuje potvrdiť stlačením tlačidla vytvoreného pre tento účel. Na virtuálnej klávesnici sú písmená abecedy rozložené podľa toho, ako často sa v anglických textoch vyskytujú[27]. Najjednoduchšie sa dajú napísať písmená, ktoré sa v textoch vyskytujú najčastejšie, naopak, písmená, ktoré sa používajú

[27] Letter frequency (http://en.wikipedia.org/wiki/Letter_frequency).

len zriedka, sa dajú napísať len pomocou viacerých dotykov. Písmená sú na virtuálnej klávesnici rozmiestené tak, aby jej používanie od užívateľa nevyžadovalo neprirodzené a nepohodlné pohyby.

4.2.4 Prepínanie režimov rozhrania

Navrhované rozhranie používa tri režimy. Prvý režim ovláda kurzor myši (režim emulácie myši), druhý ovláda funkcie jednotlivých aplikácií (režim ovládania plochy) a tretí ovláda virtuálnu klávesnicu (režim virtuálnej klávesnice). Tieto režimy môžeme prepínať dvoma spôsobmi.

Prvým spôsobom prepínania je použitie gest. Z režimu ovládania plochy a z režimu ovládania kurzoru myši môžeme zobraziť virtuálnu klávesnicu zdvihnutím rúk cez predpaženie až nad hlavu. Opačnú akciu vykonáme stlačením klávesy na ľavej časti virtuálnej klávesnice. Medzi režimom ovládania plochy a režimom emulácie myši prepíname horizontálnym gestom swipe ľavej ruky.

Druhým spôsobom prepínania je použitie hlasových povelov. Povelom *mouse* spustíme režim ovládania kurzoru myši, na povel *keyboard* sa zobrazí virtuálna klávesnica a na povel *desktop* sa zobrazí plocha.

5 Prirodzené užívateľské rozhranie KinectDesktop

Prirodzené užívateľské rozhranie *KinectDesktop* implementuje navrhované rozhranie z predchádzajúcej kapitoly. Názov rozhrania vychádza zo skutočnosti, že výsledná aplikácia ovláda plochu desktopového operačného systému a aplikácie pomocou senzoru Kinect.

V kapitolách 3.3 a 3.4 som predstavil knižnice, frameworky a vývojové nástroje, ktoré slúžia na implementáciu aplikácií pre senzor Kinect. Rozhranie KinectDesktop som implementoval pomocou vývojového nástroja Kinect for Windows SDK. Používal som prvú konečnú verziu SDK, verziu 1.0, ktorá je použiteľná aj v komerčných aplikáciách. Táto voľba ovplyvnila aj výber platformy a vývojového prostredia. Medzi softvérovými požiadavkami Kinect for Windows SDK je operačný systém Windows 7 a vývojový nástroj Visual Studio 2010. Bohužiaľ z tohto dôvodu nie je aplikácia prenositeľná, a beží len na operačných systémoch Windows 7 a vyššie. SDK podporuje tri programovacie jazyky (C++, C# a Visual Basic), z ktorých som pre implementáciu rozhrania vybral jazyk C#. Kvôli softvérovým požiadavkám som aplikáciu implementoval vo vývojovom prostredí Visual Studio 2010. Typ implementovanej aplikácie je *WPF* (*Windows Presentation Foundation*). Senzor, ktorý som použil počas implementácie a testovania rozhrania, je Kinect pre Xbox 360. Okrem knižnice SDK som pri implementácii použil ešte externú sadu knižníc *Coding4Fun Kinect Toolkit*[28].

Prirodzené užívateľské rozhranie som v predchádzajúcej kapitole rozčlenil na tri časti: ovládanie kurzoru myši, ovládanie desktopových funkcií a ovládanie textového kurzoru. Keby som implementoval tieto tri časti tak, ako som ich pôvodne navrhoval, užívateľ by musel pri ovládaní desktopu striedať tri režimy. Aby výsledné rozhranie bolo jednoduchšie a efektívnejšie, ovládanie myši a desktopových funkcií som zlúčil do jedného režimu. Výsledná aplikácia má teda dva režimy: *DesktopController* – ovládanie kurzoru myši a desktopových funkcií, a *KeyboardController* – ovládanie virtuálnej klávesnice.

Táto kapitola sa skladá z troch častí: hlavné okno rozhrania, implementácia DesktopController-u, implementácia KeyboardController-u.

5.1 Hlavné okno rozhrania KinectDesktop

Implementované rozhranie je WPF aplikácia, sledovanie užívateľa a ovládanie aplikácií je spustená pomocou hlavného okna. Pri spustení okna prebieha hľadanie pripojeného a aktívneho senzoru Kinect. Táto funkcia je implementovaná pomocou triedy *KinectSensorChooser* zo sady knižníc Coding4Fun. Táto trieda implementuje ošetrovanie chybových stavov senzoru (napr. nie je pripojený senzor, AC adaptér senzoru nie je zapojený, apod.), a v hlavnom okne aplikácie zobrazí informácie o týchto chybách. Ovláda zmeny stavu Kinectu, v prípade odpojenia senzoru inicializuje nový senzor a zastaví funkcie starého senzoru.

Rozhranie KinectDesktop používa na ovládanie aplikácií len informácie z dátového prúdu *SkeletonFrame* (dátový prúd s informáciou o aktuálnej polohe kostry užívateľa). Pri inicializácii tohto dátového prúdu je možné zmeniť parametre korekčných funkcií. Tieto funkcie slúžia na korekciu malých neúmyselných pohybov virtuálnych kĺbov. Korekcia má ale aj negatívny dopad na výkon

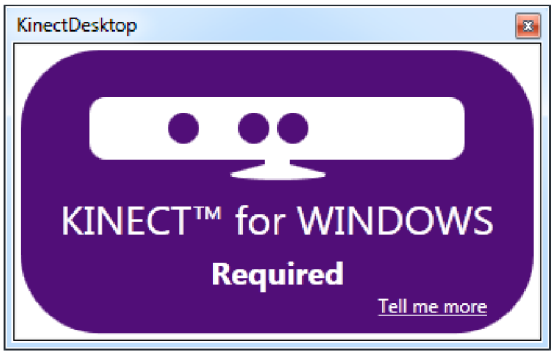
[28] Coding4Fun Kinect Toolkit (http://c4fkinect.codeplex.com/).

aplikácie. Pri testovaní týchto korekčných funkcií som najlepšie výsledky a najplynulejší pohyb virtuálnych kĺbov dostal pri kompletnom vypnutí korekcie.

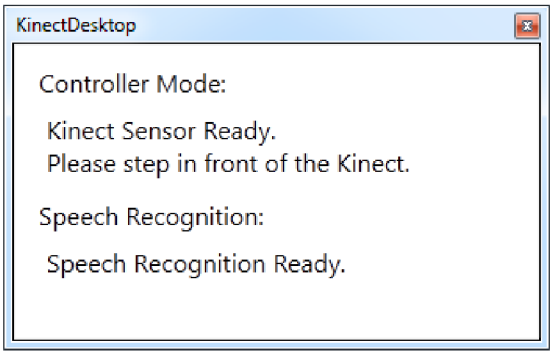
Získanie aktuálnych snímok z prúdu SkeletonFrame som implementoval pomocou udalostí. Týmto spôsobom oznámi senzor aplikácii, že aktuálna snímka je dostupná; aplikácia reaguje na túto udalosť, a obdrží aktuálnu snímku. Táto snímka obsahuje zoznam virtuálnych kostier všetkých sledovaných užívateľov. Aplikácia vyberie prvú dostupnú virtuálnu kostru, ovládanie desktopových aplikácií bude ďalej určovať pohyb jej vlastníka (užívateľa, ku ktorému patrí vybraná virtuálna kostra). Teraz už má aplikácia k dispozícii potrebné informácie o sledovanej kostre užívateľa. Informácie o kostre užívateľa sú presmerované k objektu DesktopController, alebo k objektu KeyboardController podľa aktuálneho režimu ovládania.

Pri spustení okna aplikácie sa inicializuje a odštartuje aj funkcia rozpoznávania reči. Rozhranie rozpozná slová „desktop“ a „keyboard“, ktoré slúžia na zmenu aktuálneho režimu ovládania. Inicializácia rozpoznávania reči a nastavenia rozpoznávacej gramatiky prebieha podľa štandardného postupu. Implementácia rozpoznávača reči v podstate znamenala len prispôsobenie tohto štandardného postupu k požiadavkám rozhrania KinectDesktop (implementácia rozpoznávača reči je založená na (7)).

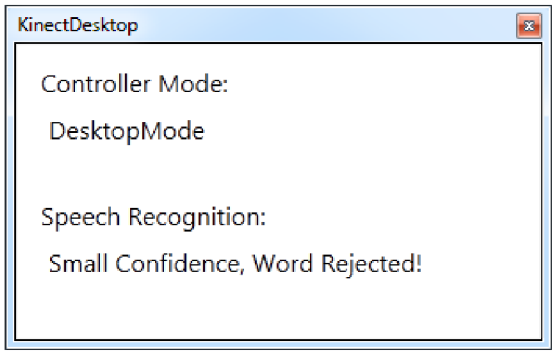
Okrem chybových hlásení triedy KinectSensorChooser nájdeme v hlavnom okne aplikácie ešte aj informácie o aktuálnom režimu ovládania (Desktop Mode alebo Keyboard Mode) a o aktuálnom stavu prebiehajúceho rozpoznávania hlasových príkazov (príkaz rozpoznaný, príkaz odmietnutý, slovo odmietnuté – nízka zhoda).



Obrázok 23 Hlavné okno – chybová správa, senzor Kinect nie je pripojený



Obrázok 24 Hlavné okno – senzor Kinect je inicializovaný, modul rozpoznávania reči načítaný



Obrázok 25 Hlavné okno - aktuálny ovládací režim je DesktopMode, rozpoznávanie príkazu nie je úspešné

5.2 DesktopController – ovládanie kurzoru myši a desktopových funkcií

Trieda DesktopController implementuje ovládanie kurzoru myši, funkcie operačného systému a aplikácie popísané v kapitole o návrhu prirodzeného užívateľského rozhrania. Užívateľ ovláda tieto funkcie pomocou statických póz a dynamických gest. Trieda obsahuje tri metódy na ovládanie funkcií vyššie zmienených:

- UpdateCursor: aktualizuje polohu myši podľa polohy virtuálneho kĺbu dlaň. Podľa pohybov ľavej ruky emuluje ľavé a pravé kliknutie myšou.
- DetectRightHandActions: metóda sleduje pohyby pravej ruky, a spustí funkcie popísané v návrhu: zatvorenie aplikácie, zobrazenie plochy, minimalizácia a maximalizácia okna, zmena okna v popredí.
- DetectLeftHandActions: pomocou tejto metódy sleduje rozhranie pohyby ľavej ruky a ovláda funkcie, ktoré sa podľa návrhu spustia pózami a gestami ľavej ruky.

Podľa návrhu prirodzeného užívateľského rozhrania je možné ovládať kurzor myši, funkcie desktopu a aplikácií len v prípade, že spôsob držania tela užívateľa splňuje isté požiadavky (napr.: ovládanie kurzoru myši – ľavá dlaň musí byť pod virtuálnym kĺbom spine). Preto pred spustením týchto metód DesktopController najprv vykoná rozpoznávanie statickej pózy užívateľa. Výsledok tohto rozpoznávania určuje, ktorá z uvedených troch metód bude s virtuálnou kostrou užívateľa pracovať.

Pretože aj tieto metódy pracujú s pózami a gestami, najprv popíšem implementáciu spôsobu rozpoznávania týchto užívateľských prejavov.

5.2.1 PoseDetector – detekcia statických póz

PoseDetector je množina tried zoskupených do menného priestoru *KinectDesktop.PoseDetection* (menný priestor - *namespace*). Tieto triedy slúžia na načítanie definícií póz zo súboru a na ich rozpoznávanie podľa vzájomnej polohy kĺbov virtuálnej kostry užívateľa.

Základná myšlienka rozpoznávania statických póz je detailne popísaná v (7). Moja implementácia je založená na rovnakých princípoch, rozširuje a vylepšuje popísané techniky. Niektoré časti zdrojového kódu sú prebrané z tohto zdroja. Zdrojové súbory s prebraným zdrojovým kódom som označil popisom autorských práv, podľa ktorých som tieto riešenia integroval do mojej práce.

Rozpoznávanie statických póz je založená na porovnaní aktuálnej virtuálnej kostry s uloženým prototypom danej pózy. Porovnáva sa vzájomná poloha istých kĺbov a veľkosť uhlu uzavretého susednými kĺbmi

Základnou triedou menného priestoru je trieda *Pose*, ktorá reprezentuje statickú pózu. Vzhľadom k dôležitosti tejto triedy pri rozpoznávaní póz, v nasledujúcich odsekoch popíšem jej tri vlastnosti:

- *poseID*: identifikátor, ktorý identifikuje, ktorú pózu definujú ostatné dve vlastnosti. Dátový typ tejto vlastnosti je *string*, dôvodom používania tohto typu sa zaoberám pri načítaní definícií póz zo súboru.
- *Angles*: zoznam objektov *PoseAngles* typu List. Tieto objekty obsahujú informáciu o uhloch medzi určitými kĺbmi.

- *Positions*: zoznam objektov *PosePosition* typu List. Tieto objekty určujú vzájomnú polohu medzi dvoma kĺbmi.

Definície statických póz, pomocou ktorých užívateľ môže ovládať funkcie aplikácií, sú uložené v súboru typu XML. Výhody tohto riešenia oproti implementácii týchto póz priamo v zdrojovom kódu sú jednoznačné: zmeniť definíciu pózy je možné bez kompilácie programu; pózy sa stávajú prenositeľným medzi aplikáciami používajúcich triedu PoseDetector; zoznam póz je jednoducho rozšíriteľný. Na načítanie definícií póz zo súboru XML slúžia metódy triedy *PoseDatabaseXML*. Objekt tejto triedy obsahuje zoznam, v ktorom sú uložené tieto načítané definície. V prípade, že by sa počas načítania a spracovania informácií zo súboru vyskytla nejaká chyba, vyvoláva sa výnimka *PoseDatabaseLoadingException*.

Vlastnosti triedy Pose sú uložené pomocou XML elementov a atribútmi. Nasledujúca časť XML súboru definuje pózu, pomocou ktorej užívateľ spustí funkciu ukázania plochy:

```
<Pose>
   <PoseID id = "ShowDesktopPose" />
   <Angles>
      <Angle centerjoint="ShoulderRight" anglejoint="ElbowRight" angle="45"
            offset="20" />
      <Angle centerjoint="ElbowRight" anglejoint="WristRight" angle="45"
            offset="20" />
   </Angles>
   <Positions>
      <Position firstjoint="HandLeft" secondjoint="Spine" poseaxle="AxleY"
            poseoperationtype="Greater" offset="0.0"/>
      <Position firstjoint="HandRight" secondjoint="Spine" poseaxle="AxleY"
            poseoperationtype="Smaller" offset="0.10"/>
   </Positions>
</Pose>
```

Obrázok 26 Póza uložená v súboru XML

Rozpoznávanie póz funguje na základe porovnávania aktuálnej polohy virtuálnej kostry so všetkými uloženými pózami. Najprv sa porovnajú uhly medzi určitými kĺbmi, potom vzájomná pozícia kĺbov podľa definície pózy. Metódu porovnania uhlov medzi kĺbmi a implementáciu vypočítania týchto uhlov som prebral z (7). Porovnanie vzájomnej polohy kĺbov som riešil porovnaním ich súradníc v súradnicovom systému Kinectu.

Po porovnaní aktuálnej polohy sledovanej kostry s uloženými pózami je výstupom PoseDetector-u buď poseID (identifikátor pózu) rozpoznanej pózy, alebo reťazec „NoPoseDetected“ – poloha kostry sa nezhoduje ani s jednou z uložených póz.

5.2.2 SwipeDetector – detekcia gesta swipe

Rozpoznávanie gesta swipe som implementoval podľa metódy, ktorá je popísaná v (7). Zvýšil som ale spoľahlivosť rozpoznania a rozšíril som funkciu detekcie gesta swipe. Moja implementácia rozpoznáva nielen horizontálne gestá užívateľa, ale aj tie vertikálne.

Trieda *SwipeDetector* slúži na rozpoznávanie horizontálnych a vertikálnych pohybov ruky užívateľa. Swipe je dynamické gesto, preto k jeho rozpoznaniu potrebujeme aj predchádzajúce polohy užívateľovej ruky: aktuálne súradnice polohy ruky sú uložené pri spracovaní každej snímky. Na ukladanie týchto informácií slúži trieda *SwipeObject*, ktorá reprezentuje prebiehajúce gesto swipe.

Objekt triedy SwipeObject obsahuje predchádzajúce súradnice dlane pravej alebo ľavej ruky a informáciu o smeru pohybu ruky.

Pre rozpoznávanie gesta swipe som stanovil nasledujúce požiadavky:

- dĺžka pohybu ruky pri vykonávaní gesta musí byť minimálne 50 cm,
- užívateľ musí vykonať celé gesto v priebehu dvoch sekúnd,
- pri vykonávaní gesta musí byť pohyb ruky priamočiary (súradnice polohy ruky počas vykonania gesta musia byť v určitom pásme).

Z užívateľských pohybov SwipeDetector rozpozná gesto swipe len v prípade splnenia týchto požiadaviek.

Rozpoznávanie horizontálneho a vertikálneho gesta ľavou rukou, a horizontálneho a vertikálneho gesta pravou rukou prebieha samostatne. Všetky tieto štyri spôsoby vykonania gesta majú vlastné objekty SwipeObject, do ktorých sú uložené súradnice ľavej a pravej dlane v každom snímku.

Vstupom algoritmu rozpoznávania gesta je jeden z týchto štyroch SwipeObject-ov, ktorý obsahuje kolekciu súradníc dlane a smer pohybu ruky. Ak posledná súradnica v kolekcii (aktuálna súradnica dlane) indikuje, že sa ruka pohybuje opačným smerom ako uložený smer pohybu objektu SwipeObject, pohyb ruky nie je gestom swipe. Pri odmietnutí pohybu SwipeObject zmaže kolekciu so súradnicami ruky a výstup rozpoznávania je správa o tom, že gesto nebolo rozpoznané. V prípade, že smer pohybu ruky odpovedá uloženému smeru, tak algoritmus kontroluje dĺžku pohybu a časový interval, do ktorého informácie o pohybu ruky patria. Ak dĺžka pohybu presahuje 50 cm a medzi uložením prvej a poslednej súradnice neuplynul čas špecifikovaný v požiadavkách, algoritmus kontroluje splnenie tretej požiadavky: porovnáva všetky súradnice z kolekcie s maximálnym a minimálnym limitom. V prípade, že sa všetky súradnice pohybujú medzi týmito hodnotami, algoritmus rozpoznal gesto, a vráti smer pohybu ruky. V opačnom prípade je pohyb opäť odmietnutí a zmaže sa kolekcia súradníc.

5.2.3 Ovládanie kurzoru myši

Ovládanie kurzoru myši som implementoval podľa návrhu popísaného v predchádzajúcej kapitole. Pohyb myši funguje len v prípade, keď ľavá dlaň je pod kĺbom spine. Súradnica kĺbu pravej dlane priamo určuje polohu kurzoru myši pomocou triedy *Cursor*[29]. Užívateľ ovláda kurzor rukou v imaginárnom štvorci, ktorý je vždy na jeho pravej strane.

Pre priame ovládanie kurzoru z imaginárneho štvorca je nutné transformovať súradnice pravej dlane zo súradnicovej sústavy Kinectu do súradnicového systému obrazovky. Knižnica Coding4Fun Kinect Toolkit obsahuje metódu *ScaleTo,* ktorá slúži na priame mapovanie súradníc zo súradnicovej sústavy Kinectu na obrazovku. Navrhnutú funkciu som implementoval modifikáciou tejto metódy:

- Ako začiatok súradnicovej sústavy Kinectu som použil virtuálny kĺb spine.
- Pomocou korekčnej hodnoty som zmenil hodnotu súradnice transformovaného kĺbu.

Aj stlačenie ľavého a pravého tlačidla myši funguje podľa návrhu popísaného v predchádzajúcej kapitole. Desať centimetrovým pohybom ľavej dlane doľava klikne užívateľ ľavým tlačidlom myši, rovnakým pohybom doprava klikne pravým tlačidlom. Implementácia

[29] MSDN Library: Cursor Class (http://msdn.microsoft.com/en-us/library/system.windows.forms.cursor.aspx).

stlačenia ľavého a pravého tlačidla myši používa funkciu v nespravovanom kóde (*unmanaged code*) *mouse_event*[30].

V návrhu rozhrania som popísal spôsob emulácie prostredného tlačidla myši. Pri testovaní som ale zistil, že navrhované riešenie je veľmi nestabilné a skoro nepoužiteľné. Dôvodom tejto nestability je podľa môjho názoru štruktúra zorného poľa Kinectu, o ktorom som sa zmienil už aj v návrhu. Vzhľadom k nepoužiteľnosti riešenia, implementáciu tejto funkcie som z mojej práce vynechal.

5.2.4 Ovládanie funkcií desktopu a aplikácií

Ovládanie desktopových funkcií je implementované podľa návrhu uvedenej v predchádzajúcej kapitole. Rozpoznávanie statických póz a dynamických gest som už predstavil v predchádzajúcich podkapitolách. Pretože na ovládanie týchto funkcií sa používajú podobné pohyby, aj implementácia detekcie týchto póz a gest v metódach DetectRightHandActions a DetectLeftHandActions je veľmi podobná:

- Ako vstup dostanú tieto metódy výsledok rozpoznávania statickej pózy. Podľa rozpoznanej pózy následne spustia požadovanú funkciu.
- Metódy odštartujú algoritmus rozpoznávania gesta swipe a na základe jeho smeru spustia požadovanú funkciu.

Viaceré funkcie som implementoval emuláciou stlačenia klávesy klávesnice, napr.: šípky, Esc alebo Enter. Na emuláciu týchto klávesov som použil metódu *SendWait* z triedy *SendKeys*[31]. Táto trieda umožňuje, aby som z môjho rozhrania posielal klávesové udalosti oknu na popredí. Funkcie operačného systému, ako minimalizácia a maximalizácia, sú ovládateľné pomocou nespravovaného kódu.

V návrhu som popísal ovládanie funkcie zmeny okna na popredí pomocou gesta swipe. Túto funkciu operačného systému som skúšal implementovať viacerými spôsobmi, avšak som vždy narazil na ťažkosti. Tieto ťažkosti spočívali v priradení správnej funkcie k už rozpoznanému gestu swipe. Žiadnym zo skúšaných spôsobov som nedosiahol výsledku, ktorý by som mohol nazvať funkčným, a tak som túto funkciu z implementácie vynechal.

5.2.5 Rozpoznávanie gesta na zmenu režimu ovládania

Gesto na zmenu režimu ovládania pozostáva zo zvislého pohybu ľavej a pravej ruky zdola nahor, pričom obe dlane musia byť nad kĺbom spine, ich horizontálna vzdialenosť nesmie byť väčšia ako 50 cm, ich vertikálna vzdialenosť má byť maximálne 10 cm.

Implementácia takýchto pohybov využíva rozpoznávanie gest a póz. Ak spôsob držania tela užívateľa odpovedá vyššie popísanému spôsobu, paralelne sa spustí detekcia gesta swipe ľavej aj pravej dlane (vzájomná poloha rúk musí v priebehu vykonania gesta zostať rovnaká). V prípade, že je rozpoznané gesto ľavej aj pravej ruky, aplikácia zmení režim ovládania.

---

[30] The mouse_event API (http://www.pinvoke.net/default.aspx/user32.mouse_event).
[31] MSDN Library: SendKeys Class
<http://msdn.microsoft.com/en-us/library/system.windows.forms.sendkeys.aspx>.

5.3 KeyboardController – ovládanie virtuálnej klávesnice

Rozhranie KinectDesktop emuluje funkcie fyzickej klávesnice pomocou virtuálnej klávesnice. Pri implementácii klávesnice podľa návrhu z predchádzajúcej kapitoly som narazil na rôzne problémy z hľadiska užívateľskej prívetivosti a efektívnosti. Hlavnými vlastnosťami, podľa ktorých sa výsledná virtuálna klávesnica hodnotí, sú rýchlosť písania, zložitosť stlačenia virtuálnych kláves, čas potrebný na zvyknutie si na jej používanie a miera spätnej väzby pri písaní.

Pri implementácii som sa inšpiroval existujúcimi riešeniami virtuálnej klávesnice: klávesnicou chytrých telefónoch, *8pen*[32] a *3D Keyboard* (18).

Implementáciu emulácie klávesnice som rozdelil na tri hlavné časti: virtuálna klávesnica a virtuálne klávesy, grafické rozhranie virtuálnej klávesnice a databáza virtuálnych znakov. V prvej časti tejto podkapitoly popíšem tieto základné súčasti KeyboardController-u, potom predstavím implementáciu emulácie funkcií klávesnice.

5.3.1 Virtuálna klávesnica a virtuálne klávesy

Obvyklým spôsobom implementácie virtuálnej klávesnice je integrácia detekcie stlačenia klávesov do grafického užívateľského rozhrania klávesnice. Poloha užívateľovej ruky je transformovaná do okna aplikácie a preberá funkciu kurzoru myši – paralelne s pohybom ruky sa v okne pohybuje aj „kurzor ruky“. Komponenty okna rozpoznajú, kedy je tento kurzor nad nimi.

Moja implementácia virtuálnej klávesnice, trieda *VirtualKeyboard*, pracuje na opačnom princípe. Grafické rozhranie klávesnice slúži len na vizualizáciu virtuálnych klávesov a polohy ruky užívateľa. Na detekciu stlačenia jednotlivých kláves potrebuje moje riešenie len aktuálnu snímku z prúdu SkeletonStream. Klávesnica funguje aj bez grafického rozhrania, čo môže byť v niektorých prípadoch výhodné. Ďalšia výhoda tohto prístupu je jeho efektivita: stlačenie klávesy je rozpoznané algoritmicky, grafické rozhranie len zobrazuje aktuálny stav virtuálnej klávesnice (stlačený kláves je zvýraznený). Zmena grafického rozhrania sa stane veľmi jednoduchou úlohou oproti prvému riešeniu, kde GUI implementuje aj vlastnú logiku klávesnice.

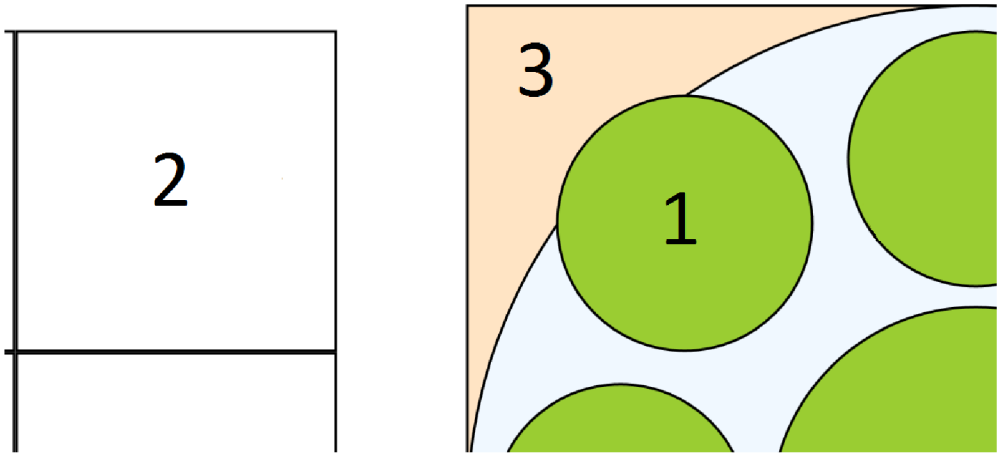
Moja virtuálna klávesnica nepracuje s hĺbkovými dátami. Na ovládanie klávesnice stačia súradnice polohy užívateľovej ruky po osách x a y. Dôvodmi, prečo som nevyužíval aj tretiu dimenziu a klávesnica je len 2D, sú nasledujúce problémy:

- Pri práci s hĺbkovými informáciami z prúdu SkeletonStream som vždy narazil na problém so štruktúrou zorného poľa senzoru Kinect. Pri pohybu ruky dopredu boli súradnice x a y nestabilné, ich hodnota nezostala nezmenená.
- Spätná väzba je pre užívateľa veľmi dôležitá, efektívna vizualizácia 3D klávesnice môže byť náročná (18).

Virtuálna klávesnica sa vždy pohybuje spolu s virtuálnym kĺbom spine užívateľovej kostry. Poloha klávesov je závislá od polohy tohto kĺbu, a sa preto ľavá časť klávesnice vždy nachádza na ľavej, a pravá časť na pravej strane užívateľa. Poloha jednotlivých kláves je aktualizovaná pri každej snímke z prúdu SkeletonStream.

[32] 8pen: Virtuálna klávesnica založená na dotykových gestách. Oficiálne stránky softvéru: <http://www.8pen.com/>.

Klávesnica pozostáva z klávesov troch typov: kruhový kláves (*VirtualCircleKey*), štvorcový kláves (*VirtualRectangleKey*) a rohový kláves (*VirtualCustomKey*). Triedy klávesov obsahujú metódu na aktualizáciu polohy klávesy, a na testovanie stlačenia klávesy. Tvar klávesov je u kruhových a štvorcových klávesov jednoznačná, tvar rohového klávesy je znázornený na Obrázku 26. Ľavá časť klávesnice pozostáva z deviatich štvorcových klávesov, na pravej časti sú štyri rohové klávesy a deväť kruhových klávesov. Prostredný kruhový kláves je väčší ako ostané, slúži na potvrdenie písania znaku.

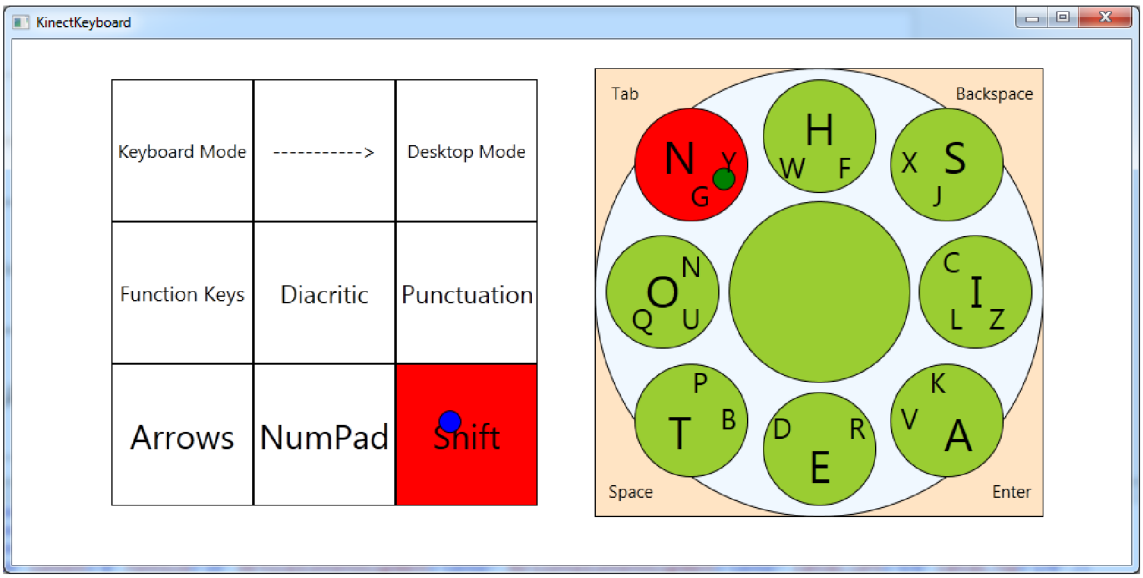


Obrázok 27 Virtuálne klávesy: 1 – kruhový, 2 – štvorcový, 3 - rohový

Trieda VirtualKeyboard má dve metódy na detekciu stlačenia klávesy, zvlášť pre ľavú a pravú stranu klávesnice. Výstupom týchto metód je identifikátor polohy klávesy, na ktorom sa kurzor pravej a ľavej ruky práve nachádza. Ak poloha kurzoru je mimo klávesov, metóda oznámi virtuálnej klávesnici, že žiadny kláves nebol stlačený.

5.3.2 Grafické rozhranie virtuálnej klávesnice

Okno grafického rozhrania virtuálnej klávesnice obsahuje rovnaké klávesy ako virtuálna klávesnica: kruhové, štvorcové a rohové klávesy. Okrem týchto tlačidiel, okno zobrazuje ešte kurzor ľavej a pravej ruky: ľavú ruku predstavuje modrá, pravú ruku zelená bodka. Súradnice polohy ľavej a pravej ruky sú transformované zo súradnicového systému Kinectu do systému okna pomocou metódy *ScaleToSpine*, ktorú som používal aj pri implementácii ovládania kurzoru myši.



Obrázok 28 Grafické rozhranie virtuálnej klávesnice: ľavá ruka stlačila klávesu *Shift*, pravou rukou je možné písať veľké písmená

Za aktualizácie obsahu okna je zodpovedná trieda *GUIManager*. Keď sa ľavou alebo pravou rukou dostanem na kláves, metódy triedy automaticky zmenia farbu tlačidla na červenú. Keď moja ruka opustí dané tlačidlo, jeho farba sa vráti späť na pôvodnú. Objekt tejto triedy ovláda aj rozloženie pravej časti klávesnice podľa stavu ľavých klávesov.

5.3.3 Databáza virtuálnych znakov

Virtuálny znak je dynamickým gestom, pri ktorom stlačím istú kombináciu virtuálnych klávesov. Pomocou takýchto gest je možné písať na pravej časti virtuálnej klávesnici. Virtuálne znaky reprezentujú objekty triedy *VirtualChar*. Táto trieda má tri vlastnosti:

- *CharType*: identifikátor, ktorým je daný znak presne určený.
- *VKeyList*: zoznam, ktorý obsahuje polohy virtuálnych klávesov, ktorých sa pri vykonaní znaku je nutné dotknúť.
- *VKeyCount*: počet objektov v zozname *VKeyList*.

Trieda má ešte metódu, ktorá porovnáva vstupný zoznam polohy klávesov so zoznamom *VKeyList*. Táto metóda je veľmi užitočná pri rozpoznávaní virtuálnych znakov.

Definície virtuálnych znakov sú načítané zo súboru typu XML. Výhoda uloženia týchto znakov spočíva v tom, že pri zmene niektorých znakov nie je nutné kompilovať celú aplikáciu a vytvorené virtuálne znaky sú prenositeľné medzi aplikáciami.

```
<VirtualChar id="Char_CL">
  <CenterKey />
  <CenterLeftKey />
  <CenterKey />
</VirtualChar><VirtualChar id="Char_BC_BR_CR">
  <CenterKey />
  <BottomCenterKey />
  <BottomRightKey />
  <CenterRightKey />
  <CenterKey />
</VirtualChar>
```

Obrázok 29 Virtuálne znaky uložené v súboru XML

Načítanie virtuálnych znakov je implementované v triede *VirtualCharDatabase*. Metóda *CheckDatabase* tejto triedy vykonáva rozpoznávanie vstupných virtuálnych znakov.

5.3.4 Emulácia funkcií klávesnice

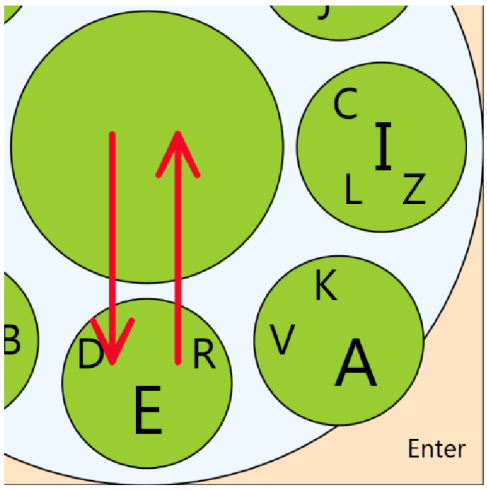
Virtuálna klávesnica rozpozná, ktoré klávesy som kurzorom ľavej a pravej ruky stlačil. V tomto kontextu stlačenie klávesy znamená dotyk kurzorom. Aktuálny stav ľavej časti klávesnice rozhoduje o rozložení pravej časti klávesnice: podľa klávesy stlačenej na ľavej časti virtuálnej klávesnice sa na jej pravej strane objavia malé alebo veľké písmená anglickej abecedy, interpunkcia, znaky s diakritikou, šípky alebo funkčné klávesy. Okrem týchto klávesov obsahuje ľavá časť klávesnice ešte tri tlačidlá, ktoré slúžia na zmenu režimu ovládania. Užívateľ ich musí stlačiť v rade za sebou, aby sa aplikácia prepla do desktopového režimu.

Pravá časť virtuálnej klávesnice sa používa na písanie písmen pomocou virtuálnych znakov. Každé písmeno má svoj virtuálny znak, pomocou ktorého je možné požadované písmeno napísať. To znamená, že aby som napísal požadované písmeno, musím vykonať jeho virtuálny znak stlačením všetkých virtuálnych klávesov, ktoré sú vo zozname znaku uvedené, a to v určitom poradí. Napríklad virtuálny znak písmena „v“ definuje nasledujúcu kombináciu klávesov: prostredný - vpravo dole - uprostred dole - prostredný. Aby som napísal písmeno „v“, musím stlačiť klávesy v tomto poradí.

Prvý aj posledný kláves vo zoznamu virtuálnych znakov je prostredný kláves, ktorý slúži na indikáciu začiatku a konca vykonávania virtuálneho znaku.

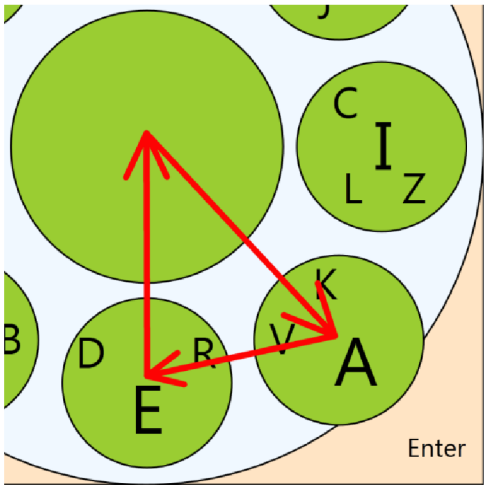
Virtuálne znaky, ktoré pri písaní používam, som rozdelil do troch skupín podľa toho, koľko klávesov musím pre vykonávanie znaku stlačiť:

- 3 klávesy: prostredný, X, prostredný



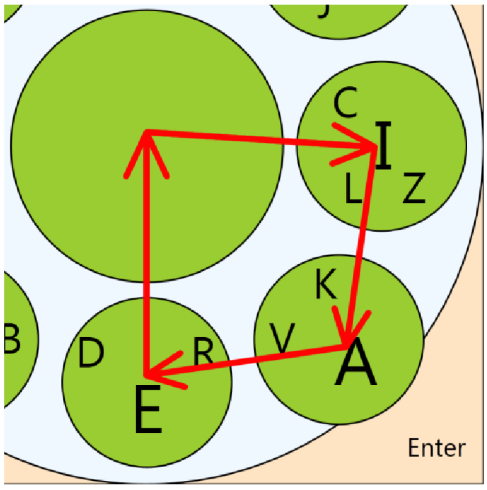
Obrázok 30 Napísanie písmena "E"

- 4 klávesy: prostredný, X, Y, prostredný



Obrázok 31 Napísanie písmena "V"

- 5 klávesov: prostredný, X, Y, Z, prostredný



Obrázok 32 Napísanie písmena "Z"

Na predvolenom rozložení pravej časti virtuálnej klávesnice sú malé písmená anglickej abecedy. Ako u návrhu ovládania textového kurzoru som naznačil, písmená sú rozložené na jednotlivé klávesy podľa frekvencie ich používania. Anglická abeceda má 26 písmen, pre napísanie týchto písmen používam rovnaký počet virtuálnych znakov. Pri písaní najčastejšie používaných

písmen sa užívateľ musí okrem počiatočného a koncového stlačenia prostredného klávesu dotknúť len jedného klávesu.

6 Výsledky a vyhodnotenie práce

Výsledkom mojej práce je prirodzené užívateľské rozhranie, ktoré emuluje funkciu klávesnice a myši pomocou senzoru Kinect. Umožňuje priame ovládanie počítača a desktopových aplikácií použitím statických póz, dynamických gest a hlasových povelov.

Implementované rozhranie môžem nazvať prirodzeným, pretože:

- na jeho ovládanie sa používajú pohyby, ktoré užívateľ bežne používa pri každodennej interakcii so svetom, a tak nie je potrebné, aby sa naučil niečo nové. Pohyby pre ovládanie som vybral tak, aby boli navyše aj jednoduché a pohodlné.
- na detekciu a sledovanie užívateľa používa len senzor Kinect, bez použitia obvyklých hardvérov ako je myš a klávesnica.

Emulácia funkcie myši: Moje rozhranie emuluje pohyb kurzoru myši, ľavý a pravý klik. Pohyb kurzoru priamo súvisí s pohybom pravej dlane užívateľa, na kliknutie tlačidlami sa používa ľavá ruka. V návrhu rozhrania som uviedol aj možný spôsob emulácie prostredného tlačidla myši, avšak do implementácie som to už nezahrnul. Dôvody som uviedol tak pri návrhu, ako aj pri implementácii rozhrania.

Emulácia funkcie klávesnice: Emulácia klávesnice je skoro úplná, virtuálna klávesnica totiž obsahuje skoro každý kláves štandardnej klávesnice (neobsahuje napr. klávesy F1 až F12, CapsLock, ScrollLock, apod.). Písmená na virtuálnej klávesnici som rozmiestnil podľa ich frekvencie použitia v anglických textoch. Frekventované písmená sa dajú napísať menším počtom pohybov, než písmená, ktoré sa vyskytujú menej často.

Možné problémy pri používaní rozhrania:

- Keď sú virtuálne kĺby veľmi blízko sebe, alebo sa prekrývajú, môže sa stať, že ich Kinect nerozpozná, a prestane ich sledovať. Toto môže nastať najčastejšie v prípade, že sa užívateľova ruka dostane pred jeho trup alebo hlavu. Následkom toho je prerušovaný pohyb kurzora.
- Pri testovaní rozhrania som zistil, že napriek tomu, že ku kliknutiu je potrebný len veľmi malý pohyb ľavej ruky, nemusí toto riešenie byť pohodlné pre každého užívateľa.
- Taktiež som zistil, že umiestniť kurzor myši na menšie tlačidlo môže byť obtiažne.
- Implementovaná virtuálna klávesnica je síce reálne použiteľná na písanie textov, avšak jej použitie je oveľa pomalšie než písanie na klasickej klávesnici, a to aj potom, čo si užívateľ na virtuálnu klávesnicu už zvykol.
- Pri testovaní taktiež vyšlo najavo, že na počítači s menším výkonom môže byť pohyb kurzoru neplynulý (testy prebiehali na procesoroch AMD FX-8120 a AMD Turion X2 ZM-82).

Možnosti ďalšieho vývoja rozhrania: Vzhľadom na to, že technológia, ktorú som v tejto práci používal je relatívne nová (Kinect je dostupný od 4. novembra 2010, verziu 1.0 vývojového nástroja Kinect for Windows SDK vydali 1. februára 2012) a taktiež veľmi zaujímavá, myslím si, že existuje v podstate nekonečné množstvo možností na jej ďalší vývoj. Na záver tejto kapitoly uvediem štyri možnosti vývoja, ktoré osobne považujem za reálne a užitočné:

- Prispôsobenie používania rozhrania na používanie ľavákmi: možnosť výmeny pohybov pravej a ľavej ruky.

- Pridanie možnosti zmeny vzhľadu grafického rozhrania (rôzne veľkosti a rozmiestnenie tlačidiel, rôzne farebné schémy, apod.).
- Implementácia rozhrania na ovládanie desktopu založeného na princípu, ktorý som popísal v podkapitole 4.1.4 (implementácia rozhrania podobného projektu Put That There).
- Zmena rozhrania tak, aby virtuálna klávesnica a myš boli použiteľné súčasne – klávesnica ľavou rukou, myš pravou.

Na obrázkoch 32 a 33 sú uvedené príklady použitia textového editoru.

Obrázok 33 Predvolený stav virtuálnej klávesnice - malé písmená anglickej abecedy

Obrázok 34 Režim šípok

7 Záver

Cieľom mojej diplomovej práce bolo vytvoriť rozhranie, ktoré emuluje funkcie klávesnice a myši pomocou senzoru Kinect. Myslím si, že moje riešenie tejto úlohy je úspešné a splňuje menované kritériá. Až na pár málo výnimiek je rozhranie efektívne a reálne použiteľné, pre užívateľa pohodlné a prívetivé. Okno virtuálnej klávesnice je prehľadné, spôsob jej používania je jasné už pri prvom pohľade.

Pred zahájením práce som sa musel oboznámiť s teoretickými základmi prirodzených užívateľských rozhraní. Musel som pochopiť hlavné princípy ich funkcie, a musel som vymyslieť, ako tieto princípy integrovať do praktickej časti mojej diplomovej práce. Ďalej som sa musel zoznámiť so zariadením, ktoré som pri práci používal, so senzorom Kinect. Nakoniec som sa musel zoznámiť aj s programovacím jazykom C# a vývojovým nástrojom Kinect for Windows SDK. Len po získaní týchto vedomostí som bol schopný vytvoriť rozhranie, ktoré splňuje požiadavky prirodzeného užívateľského rozhrania a zároveň používa informácie zo senzoru Kinect k ovládaniu desktopových aplikácií. Podľa môjho názoru je technológia Kinectu veľmi perspektívna, a rád by som sa jej venoval aj v budúcnosti. Získané vedomosti by som tak mohol uplatniť pri práci na podobných projektoch.

V priebehu vypracovania praktickej časti mojej práce som získal mnoho skúseností tak v navrhovaní rozhrania, ktoré je pre väčšinu užívateľov prívetivé, ako aj s prácou so senzorom Kinect. Pri práci s týmto hardvérom som narazil na niekoľko ťažkostí, ktoré v istej miere spomalili moju prácu. Najväčšie problémy mi spôsobili nepresnosti senzoru Kinect pri sledovaní virtuálnej kostry užívateľa. Myslím si ale, že tento problém bude pri vývoji ďalších verzií tohto senzoru eliminovaný, čo uľahčí a spríjemní jej použitie v rámci rôznych projektov.

Ako každá aplikácia, samozrejme aj moje rozhranie obsahuje chyby, ktoré som už popísal v predchádzajúcej kapitole. Vzhľadom k tomu, že použitá technológia je pomerne nová, skrýva v sebe veľa možností ďalšieho vývoja, v rámci ktorého môžu byť tieto chyby odstránené.

Myslím si, že pri súčasnom tempe vývoja informačných technológií ovládanie počítačov bez použitia akýchkoľvek vstupných zariadení sa čoskoro stane reálnou záležitosťou aj v každodennom živote bežných užívateľov.

Literatúra

1. **Hewett; Baecker; Card; Carey; Gasen; Mantei; Perlman; Strong; Verplank.** Curricula for Human-Computer Interaction. *ACM SIGCHI.* [Online] 29. 7 2006. [Dátum: 22. 5 2012.] http://old.sigchi.org/cdg/index.html.

2. **Landauer, Thomas K.; Helander, Martin G.; Prabhu, Prasad V.;.** *Handbook of Human-Computer Interaction.* Amsterdam : North-Holland, 1997. 9780444818621.

3. **Raymond, Eric Steven; Landley, Rob W.;.** A Brief History of User Interfaces. *The Art of Unix Usability.* [Online] 18. 4 2004. [Dátum: 22. 5 2012.] http://www.catb.org/~esr/writings/taouu/html/ch02.html.

4. *A Brief History of Human Computer Interaction Technology.* **Myers, Brad A.** 2, March 1998, ACM interactions, Zv. 5, s. pp. 44-54.

5. **Group, Bryan.** Metro UI – 21st Century Revolutionist. *MSDN Blogs.* [Online] 9. 2 2012. [Dátum: 22. 5 2012.] http://blogs.msdn.com/b/bryang/archive/2012/02/09/metro-ui-21st-century-revolutionist.aspx.

6. **Blake, Joshua.** *Natural User Interfaces in .NET.* s.l. : Manning Publications, 2012. 9781935182818.

7. **Webb, Jarrett; Ashley, James;.** *Beginning Kinect Programming with the Microsoft Kinect SDK.* s.l. : Apress, 2012. 9781430241041.

8. **Bolt, Richard A.** "Put-That-There": Voice and Gesture at the Graphics Interface. *MIT Media Lab.* [Online] 1980. [Dátum: 22. 5 2012.] http://www.media.mit.edu/speech/papers/1980/bolt_SIGGRAPH80_put-that-there.pdf.

9. **Garg, Pragati; Aggarwal, Naveen; Sofat, Sanjeev;.** Vision Based Hand Gesture Recognition. [Online] 2009. [Dátum: 22. 5 2012.] http://imageprocessinggroup.googlecode.com/svn/trunk/STICKTEMP/handTrackling/v49-173.pdf.

10. **Murthy, G. R. S.; Jadon, R. S.;.** A REVIEW OF VISION BASED HAND GESTURES RECOGNITION. *Computer Science & Electronics Journals.* [Online] 2009. [Dátum: 22. 5 2012.] http://www.csjournals.com/IJITKM/PDF/34-G.R.S.Murthy_R.S.Jadon.pdf.

11. **Zhou, Ren; Jingjing, Meng; Junsung, Yuan; Zhengyou, Zhang;.** Robust hand gesture recognition with kinect sensor. *ACM Digital Library.* [Online] 2011. [Dátum: 22. 5 2012.] http://dl.acm.org/citation.cfm?doid=2072298.2072443.

12. **Kurtenbach, Gord a Hulteen, Eric.** Gestures in Human-Computer Communication. *The Art and Science of Interface Design0990.* s.l. : Addison-Wesley Publishing Co.

13. **Zabulist, X, Baltzakis, H a Argyros, A.** Vision-based Hand Gesture Recognition. *Foundation for Research and Technology - Hellas (FORTH).* [Online] 6 2009. [Dátum: 22. 5 2012.] http://www.ics.forth.gr/~argyros/mypapers/2009_06_book_hci_gestures.pdf.

14. ElevationAngle Property. *MSDN Library.* [Online] [Dátum: 22. 5 21012.]

15. **Kean, Sean; Hall, Jonathan; Perry, Phoenix;.** *Meet the Kinect: An Introduction to Programming Natural User Interfaces* . s.l. : Apress, 2011. 9781430238881.

16. **Reichinger, Andreas.** Kinect Pattern Uncovered. [Online] 3. 4 2011. [Dátum: 22. 5 2012.] http://azttm.wordpress.com/2011/04/03/kinect-pattern-uncovered/.

17. **OpenNI.** Programmer Guide. *OpenNI Documentation.* [Online] [Dátum: 22. 5 2012.] http://openni.org/Documentation/ProgrammerGuide.html.

18. *VIRTUAL 3D KEYBOARD FOR GESTURE BASED TYPING.* **Jetenský, Pavel.** Liberec : IMEA, 2011. Sborník příspěvků z XI. mezinárodní konference IMEA 2011. 9788073727208.

Zoznam obrázkov

Obrázok 1 Približné časové údaje reprezentujú vývoj jednotlivých technológii (4) 5
Obrázok 2 Put That There (8) 10
Obrázok 3 CyberGlove II (zdroj obrázku) 11
Obrázok 4 Leon Theremin pri hre na vlastnom nástroji (zdroj obrázku) 11
Obrázok 5 Metóda rozpoznávania dynamických gest (9) 12
Obrázok 6 Kalkulátor ovládaný ručnými pózami (11) 14
Obrázok 7 Microsoft Kinect for Xbox 360 18
Obrázok 8 AC adaptér potrebný k pripojeniu Kinectu k počítaču (15) 19
Obrázok 9 Umiestnenie jednotlivých senzorov na Kinectu 19
Obrázok 10 Výpočet hĺbkovej informácie skúmaných objektov v priestoru (15) 20
Obrázok 11 Referenčný dizajn PrimeSense senzoru 22
Obrázok 12 Vrstvový model štruktúry OpenNI (17) 24
Obrázok 13 Rozdiel medzi normálnym a blízkym režimom senzoru Kinect for Windows 25
Obrázok 14 Hĺbkový obráz z Kinectu, na hornej lište je vzdialenosť objektu od senzora (7) 27
Obrázok 15 Kostra užívateľa v Kinect for Windows SDK: kostra pozostáva z 20 kĺbov 27
Obrázok 16 Virtuálna kostra užívateľa, a uhly medzi kĺbmi a kosťami virtuálnej kostry užívateľa (7) 32
Obrázok 17 Poloha zápästia voči lakte počas predvedenia gesta mávanie rukou (7) 33
Obrázok 18 Zorné pole senzoru Kinect (7) 36
Obrázok 19 Zatvorenie okna 37
Obrázok 20 Ukázať plochu 37
Obrázok 21 Kláves Esc 38
Obrázok 22 Ponuka Štart 38
Obrázok 23 Hlavné okno – chybová správa, senzor Kinect nie je pripojený 41
Obrázok 24 Hlavné okno – senzor Kinect je inicializovaný, modul rozpoznávania reči načítaný 41
Obrázok 25 Hlavné okno - aktuálny ovládací režim je DesktopMode, ... 41
Obrázok 26 Póza uložená v súboru XML 43
Obrázok 27 Virtuálne klávesy: 1 – kruhový, 2 – štvorcový, 3 - rohový 47
Obrázok 28 Grafické rozhranie virtuálnej klávesnice: ľavá ruka stlačila klávesu *Shift*, pravou rukou je možné písať veľké písmená 48
Obrázok 29 Virtuálne znaky uložené v súboru XML 49
Obrázok 30 Napísanie písmena "E" 50
Obrázok 31 Napísanie písmena "V" 50
Obrázok 32 Napísanie písmena "Z" 50
Obrázok 33 Predvolený stav virtuálnej klávesnice - malé písmená anglickej abecedy 53

Obrázok 34 Režim šípok 53

Príloha 1

Hardvérové a softvérové požiadavky Kinect for Windows

Hardvérové požiadavky Kinect for Windows SDK:

- 32 bit (x86) alebo 64 bit (x64) procesor
- Dvojjadrový procesor, s minimálnou frekvenciou 2,6 GHz
- Priradená USB zbernica
- 2 GB RAM
- Kinect for Windows/Kinect for Xbox senzor s AC adaptérom.

Softvérové požiadavky Kinect for Windows SDK:

- Operačný systém Windows 7, Windows Embedded Standard 7 alebo Windows 8
- .NET Framework 4.0
- Microsoft Visual Studio 2010 Express alebo vyššie edície
- Microsoft Speech Platform SDK v11 – vývoj aplikácie s rozpoznávaním reči

Príloha 2

Užívateľská príručka – KinectDesktop

Softvérové požiadavky aplikácie:

- Operačný systém Windows 7 alebo vyššie
- .NET Framework 4.0
- Kinect for Windows Runtime alebo Kinect for Windows SDK

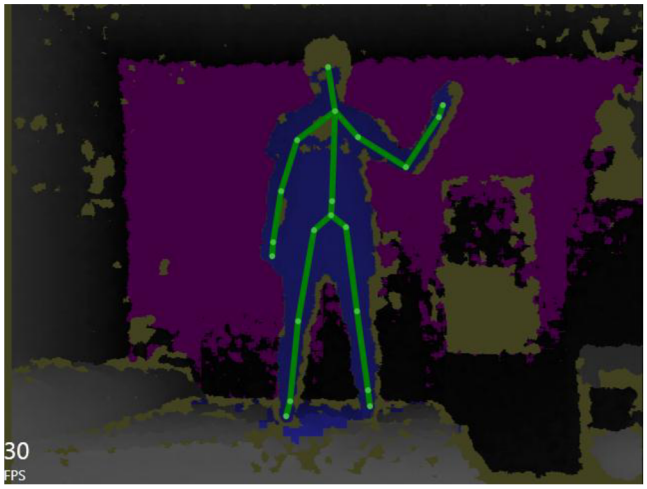
Spustenie aplikácie:

Aplikácia sa spustí pomocou príkazu *KinectDesktop.exe*. K použitiu aplikácie KinectDesktop je potrebné k počítači pripojiť senzor Kinect. V prípade, že je senzor riadne pripojený k počítači, sa na obrazovke objaví hlavné okno aplikácie. V opačnom prípade sa v tomto okne objaví chybové hlásenie *Kinect for Windows Required* a aplikácia sa môže používať až po pripojení senzoru.

Použitie aplikácie:

Aplikácia má dva režimy: režim ovládania desktopu a režim ovládania virtuálnej klávesnice; pri spustení aplikácie sa automaticky aktivuje režim ovládania desktopu.

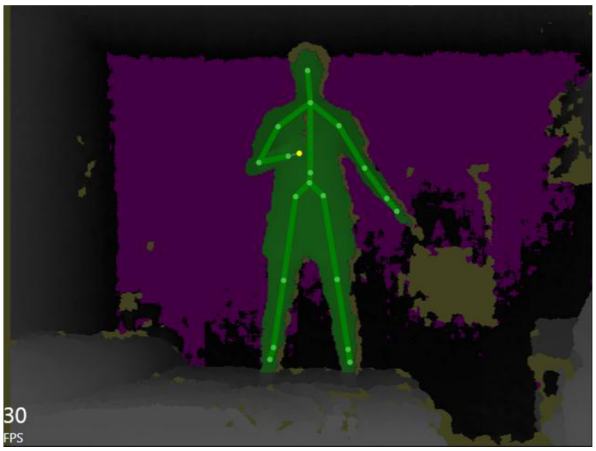
1. Ovládanie desktopu:
    a. <u>Ovládanie kurzoru myši</u>: K ovládaniu kurzoru myši sa používajú obe ruky, pravá ruka je zdvihnutá, ľavá ruka voľne vedľa tela. Samotný pohyb kurzoru sa ovláda pohybom pravej dlane, na kliknutie sa používa malý pohyb spustenej ľavej ruky doľava alebo doprava (pohyb doľava ľavý klik, pohyb doprava pravý klik).



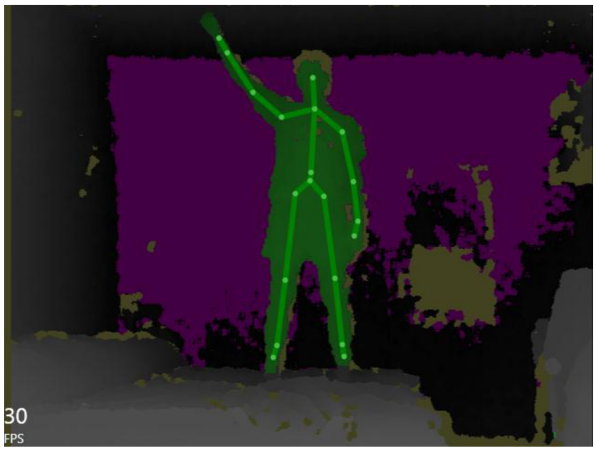
b. Pózy a gestá na ovládanie funkcii desktopu:

- Zatvorenie okna: pravá ruka je vystretá a s telom uzatvára uhol 135°. Ľavá dlaň je medzi ľavým ramenom a pupkom. Pred vykonaním ďalšieho príkazu je potrebné pravou rukou pripažiť.

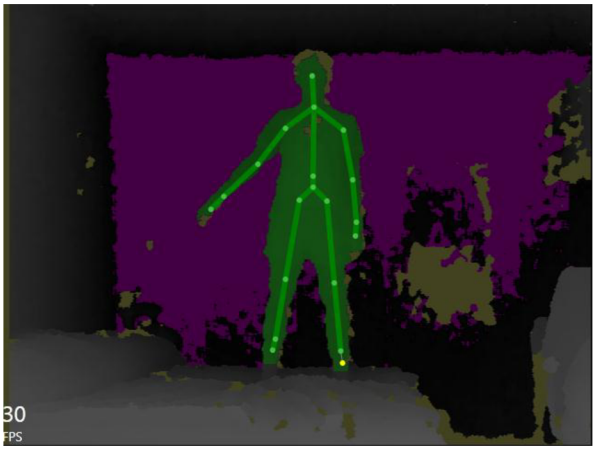
- Zobrazenie plochy: pravá ruka je vystretá, s telom uzatvára uhol 45°. Ľavá dlaň je medzi ľavým ramenom a pupkom. Pred vykonaním ďalšieho príkazu je potrebné pravou rukou pripažiť.



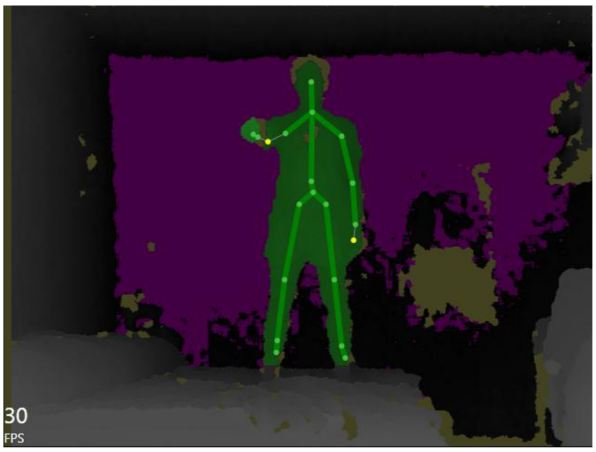
- Stlačenie klávesy Esc: pravá ruka v pripažení, ľavá ruka je vystretá, uhol medzi ľavou rukou a telom je 135 stupňov. Pred vykonaním ďalšieho príkazu je potrebné ľavou rukou pripažiť.



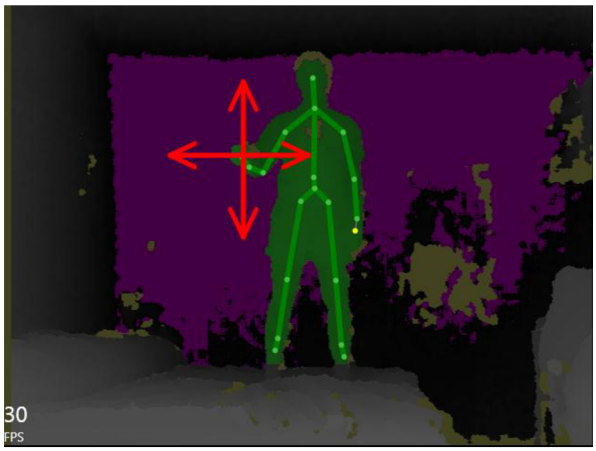
- Spustenie ponuky Štart: pravá ruka v pripažení, ľavá ruka je vystretá, uhol medzi ľavou rukou a telom je 45 stupňov. Pred vykonaním ďalšieho príkazu je potrebné ľavou rukou pripažiť.



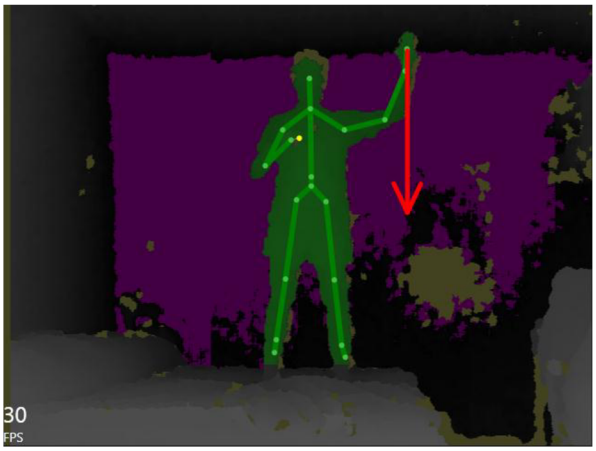
- Stlačenie klávesy Enter: pravá ruka je v pripažení, ľavá ruka v predpažení.



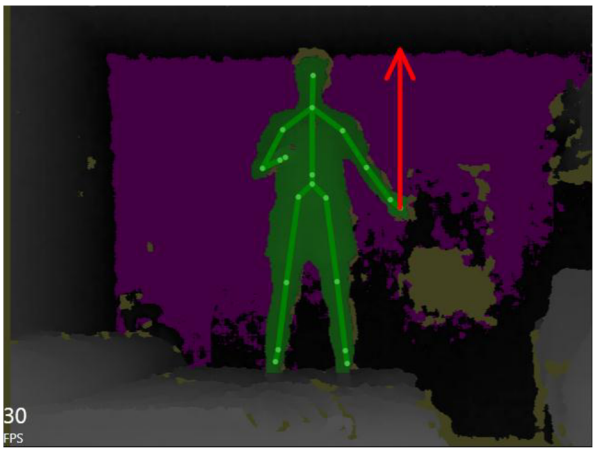
- Ovládanie šípok: pravá ruka v pripažení: smer pohybu ľavej dlane sa zhoduje so smerom šípky na požadovanej klávese.



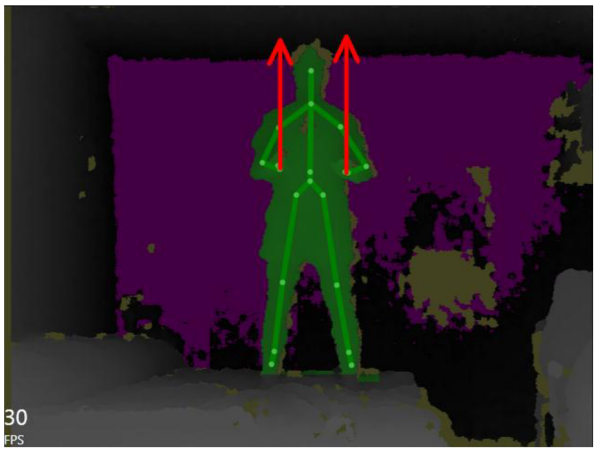
- Minimalizácia okna na popredí: pravá dlaň sa pohybuje smerom zdola nahor. Ľavá dlaň je medzi ľavým ramenom a pupkom.



- Maximalizácia okna na popredí: pravá dlaň sa pohybuje smerom zdola nahor. Ľavá dlaň je medzi ľavým ramenom a pupkom.

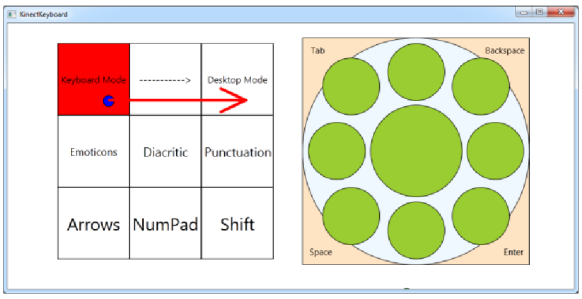


c. Zmena režimu: Prepnutie z režimu ovládanie desktopu do režimu virtuálnej klávesnice je možné súčasným pohybom oboch dlaní od pupka smerom nahor. Dlane musia byť blízko sebe a musia sa pohybovať naraz.



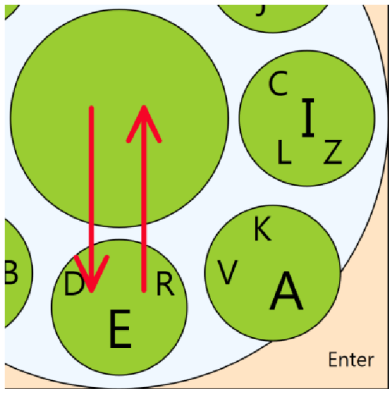
2. Ovládanie virtuálnej klávesnice: K ovládaniu virtuálnej klávesnice sa používajú obe ruky, ľavú ruku v okne predstavuje modrá, pravú ruku zelená bodka. Klávesnica je rozdelená na dve hlavné časti. Na jej pravej strane sa nachádzajú klávesy so znakmi, jej ľavou časťou sa ovláda rozloženie týchto znakov.
    a. Spôsoby rozloženia klávesov:
        - malé písmená anglickej abecedy (predvolené)
        - veľké písmená anglickej abecedy
        - numerická klávesnica
        - šípky
        - interpunkcia
        - znaky s diakritikou (slovenská abeceda)
        - emotikony

Ľavou časťou klávesnice sa ovláda aj funkcia zmeny režimu ovládania (z režimu virtuálnej klávesnice do režimu ovládania desktopu) a to stlačením troch horných klávesov za sebou, zľava doprava.

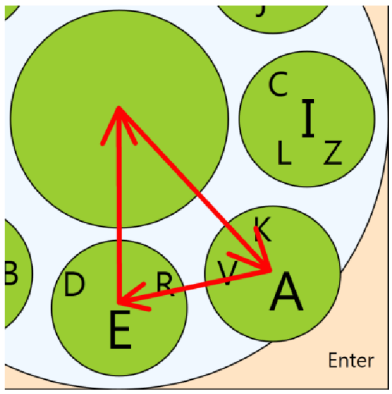


b. Písanie znakov: Znaky sa píšu plynulými pohybmi pravej dlane, písanie každého znaku sa začína a končí stlačením prostredného klávesy.

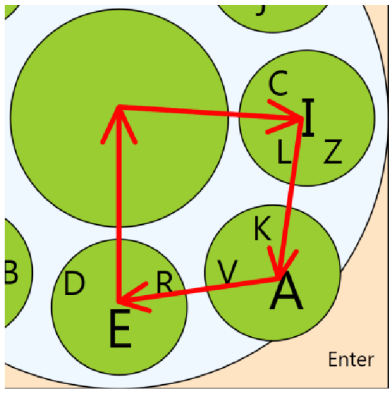
Prvý obrázok znázorňuje spôsob napísania písmena „E“: prostredný kláves – kláves so znakom „E“ – potvrdenie znaku prostredným klávesom.



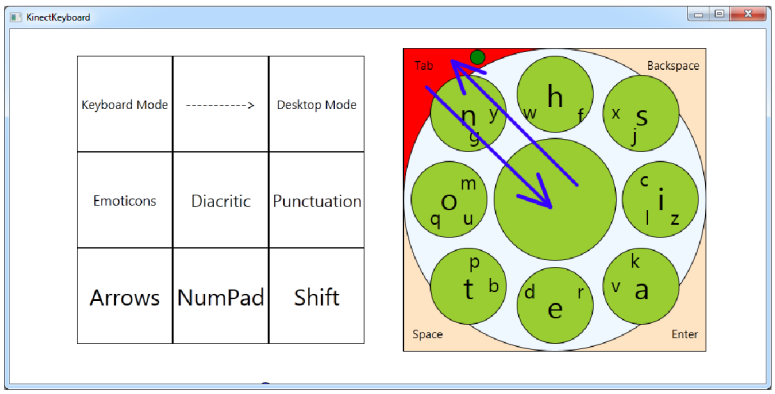
Napísanie písmena „V“. Keď sa na klávese nachádza viac písmen, a chceme napísať niektorý zo znakov znázornených menším písmom, postupujeme takto: z prostrednej klávesy prejdeme kurzorom najprv na klávesu s požadovaným znakom, potom sa dotkneme susedného klávesu vo smeru tohto znaku a nakoniec potvrdíme prostredným tlačidlom.



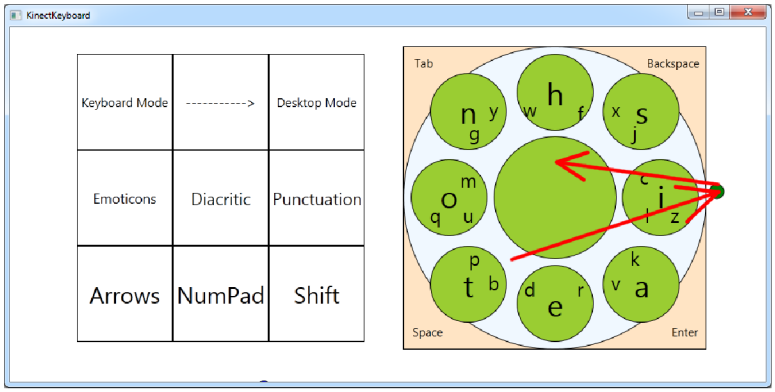
V prípade, že sa na klávese nachádzajú štyri znaky, postupujeme podobne, ako v predchádzajúcom prípade. Jedinou zmenou je, že sa dotkneme dvoch susedných klávesov vo smeru písaného znaku. Na obrázku je znázornený postup napísania písmena „Z“.



Klávesy *Tab*, *Backspace*, *Enter* a *Space* sa používajú rovnako.



Pri chybe vzniknutej pri písaní znakov stačí, aby sa zelený kurzor pravej ruky dostal mimo štvorec klávesnice. Aplikácia preruší aktuálny príkaz, a je možné pohyb pre napísanie znaku opakovať.



Zoznam príloh

Príloha 1. Hardvérové a softvérové požiadavky Kinect for Windows SDK
Príloha 2. Užívateľská príručka – KinectDesktop
Príloha 3. CD so zdrojovými kódmi aplikácie KinectDesktop a s elektronickou podobou technickej správy