Mendelova univerzita v Brně
Provozně ekonomická fakulta

Inteligentná pracovná plocha

Diplomová práca

Vedúci práce:
Ing. David Procházka, Ph.D.

Bc. Jakub Kozák

Brno 2017

Na tomto mieste chcem poďakovať vedúcemu práce Ing. Davidovi Procházkovi, Ph.D. za ústretovosť, vecné poznámky a motiváciu pri písaní tejto práce. Ďakujem taktiež svojej rodine a priateľke za neustálu podporu a trpezlivosť.

Čestné prehlásenie

Prehlasujem, že som prácu: **Inteligentná pracovná plocha**
vypracoval/a samostatne a všetky použité zdroje a informácie uvádzam v zozname použitej literatúry. Súhlasím, aby moja práca bola zverejnená v súlade§ 47b zákona č. 111/1998 Sb., o vysokých školách v znení neskorších predpisov a v súlade s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Som si vedomý, že sa na moju prácu vzťahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzatvorenie licenčnej zmluvy a použitie tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.
Ďalej sa zaväzujem, že pred spísaním licenčnej zmluvy o použití diela inou osobou (subjektom) si vyžiadam písomné stanovisko univerzity, že predmetná licenčná zmluva nie je v rozpore s oprávnenými záujmami univerzity a zaväzujem sa uhradiť prípadný príspevok na úhradu nákladov spojených so vznikom diela, a to až do ich skutočnej výšky.

V Brne dňa 19.5.2017 .....

Abstract

Kozák, J. Smart workspace. Master thesis. Brno, 2017

Interactive collaborative enviroments are solved by many companies and research teams. This thesis is focused on design and development of own system for gesture recognition based on Microsoft Kinect controller. The system is focused on collaboration of multiple users. Therefore, proposed solution can distinguish interactions of different users. The result allows to create an interactive area from a common wall using a computer, projector and Kinect controller.

Keywords

Microsoft Kinect, interaction, gesture, projector, wall, collaboration

Abstrakt

Kozák, J. Inteligentná pracovná plocha. Diplomová práca. Brno, 2017

Interaktívne kolaboratívne riešenia sú aktuálnou problematikou, ktorou sa zaoberá množstvo spoločností, či ľudí pracujúcich v tejto oblasti. Táto práca sa venuje návrhu a implementácii vlastného systému na detekciu gest s použitím senzora Microsoft Kinect. Systém je následne otestovaný na interakciu s viacerými užívateľmi na aplikácii implementovanej za týmto účelom. Vytvorené riešenie dokáže detekovať dotyky pre jedného, aj pre viacerých užívateľov. Výsledok tejto práce umožňuje z obyčajnej steny vytvoriť interaktívnu a to s pomocou počítača, projektora a senzora Microsoft Kinect.

Kľúčové slová

Microsoft Kinect, interakcia, gesto, projektor, stena, kolaborácia

Obsah

1 Úvod

S interaktívnymi multimediálnymi prvkami sa stretávame čoraz častejšie. Či už je to v školách, obchodoch, múzeách, dokonca aj na verejných priestranstvách. S príchodom dotykových mobilných telefónov sa interaktívne a hlavne dotykové prvky objavujú takmer všade.

Takéto interaktívne prvky sú výborným prostriedkom pre kolaboratívnu prácu. Pod pojmom kolaboratívna práca si nemusíme predstaviť veľa ľudí v jednej miestnosti, dokonca pri jednej tabuli, ktorí riešia spoločnú úlohu. Takáto práca môže fungovať aj na diaľku. Dobrým príkladom je *Google Drive*, ktorý ponúka množstvo kolaboratívnych nástrojov, ktoré sú podobné nám známym ako *Word*, *Excel* a podobne. Tu je nutné spomenúť taktiež *Google Hangouts*, ktorý je v tomto smere silným nástrojom.

Čo sa týka kolaborácie v jednej miestnosti, prevažujú interaktívne tabule. Interaktívne tabule, či rôzne iné kolaboratívne prostredia bezpochybne zvyšujú efektívnosť výučby v školách. Nie je to všetko iba o školách, kolaboratívne prostredia sa veľmi často vyskytujú aj v pracovnom prostredí. Ide o jednoduchý a pritom veľmi účinný spôsob ako zvýšiť produktivitu, či už žiakov, alebo zamestnancov. Aj to je dôvodom, že sa touto oblasťou zaoberá veľké množstvo spoločností a snažia sa pre tento trh poskytnúť čo najlepší produkt.

Existuje veľké množstvo rôznych typov takýchto produktov. Či už sa jedná o menšie dotykové plochy, veľké dotykové plochy alebo hlasovacie zariadenia. Všetky majú spoločnú vlastnosť a to prehľadná prezentácia dát pre všetkých účastníkov, možnosť rýchlej interakcie, prípadne interakcie viacerých pracujúcich zároveň.

Aj známa spoločnosť *Microsoft* stále pracuje na takomto interaktívnom zariadení. Nesie názov *Microsoft Surface Hub* a jedná sa o veľkú dotykovú obrazovku s množstvom funkcií. Určený je hlavne pre pracovné prostredia, keďže cena menšieho modelu presahuje 200 000 Kč.

Cena je veľakrát dôvodom, prečo sa firmy, prípadne rôzne organizácie k takémuto riešeniu nepriklonia. Preto vznikajú na trhu produkty, ktoré sa snažia znížiť náklady na vytvorenie takéhoto interaktívneho prostredia. Väčšinou je nižšia cena dosiahnutá použitím lacnejšiej technológie, ktorá zabezpečuje interakciu užívateľa. To má však samozrejme vplyv na výslednú kvalitu systému. Záleží iba na konkrétnom zákazníkovi, na čo systém potrebuje, aké plánuje jeho využitie a aký má rozpočet.

Tieto skutočnosti naznačujú, že sa jedná o aktuálnu problematiku, ktorou sa zaoberá stále viac firiem a ľudí, ktorí v tejto oblasti pracujú. Aj preto som sa rozhodol vypracovať túto prácu, v ktorej sa pokúsim navrhnúť nízkorozpočtový systém, ktorý by sa aspoň približoval hotovým riešeniam, ktoré súčasný trh ponúka.

2 Cieľ práce

Cieľom tejto práce je navrhnúť koncept inteligentnej pracovnej plochy s použitím senzora *Microsoft Kinect* a implementovať aplikáciu, pomocou ktorej bude jeho funkčnosť overená. K splneniu tohto hlavného cieľa bude potrebné vykonať viacero postupných krokov.

Prvým krokom bude vytvoriť si prehľad o tejto problematike. Tento krok zahŕňa získať informácie o rozličných typoch interaktívnych plôch a o princípe, na akom fungujú. Dôležitá je taktiež analýza súčasného trhu, ktorá prinesie informácie o tom, či niekto takéto riešenia ponúka, prípadne za akú cenu a podobne. Analýza neostane len pri komerčných riešeniach, preskúmané budú aj projekty, ktoré sa zaoberali podobnou problematikou.

Vrámci druhého kroku bude vykonaná taktiež analýza a to v oblasti hĺbkových senzorov so zameraním na *Microsoft Kinect*. Táto analýza nebude rozsiahla a to z dôvodu, že hĺbkový senzor je vopred vybraný, konkrétne spomínaný *Microsoft Kinect*. Je však dôležité porovnať tento senzor s inými senzormi a prípadne navrhnúť budúce riešenie s iným senzorom. K tomuto kroku bude patriť aj preskúmanie existujúcich frameworkov, ktoré umožňujú prácu so senzorom *Microsoft Kinect*.

Stanovenie metodiky bude tretím krokom k dosiahnutiu hlavného cieľa. V metodike bude stanovené umiestnenie senzora, programovací jazyk, vývojové prostredie a taktiež framework pre prácu so senzorom. Spomínaný výber je nevyhnutý pre začiatok praktickej časti tejto práce.

Štvrtým krokom bude návrh a implementácia systému, ktorý sa bude odvíjať od informácií, ktoré budú získané počas vykonávania rešerše.

Návrh a implementácia aplikácie, ktorá otestuje navrhnutý a implementovaný systém bude piatym čiastkovým krokom.

Posledným krokom bude zhodnotenie vytvoreného systému.

3 Rešerš

3.1 Rozličné typy interaktívnych plôch

Vo všeobecnosti sa interaktívna plocha skladá z troch častí. Prvou z nich je zobrazovacia časť. Môže sa jednať o projektor, alebo aj priamo monitor. Druhou z nich je výpočtová jednotka, najčastejšie bežný počítač, ktorý ma na starosti zobrazovaný obsah ako aj interakciu užívateľa. Treťou a najdôležitejšou časťou je zariadenie, resp. systém, ktorý zabezpečuje interakciu užívateľa s takouto stenou. Podrobnejšie sa pozrieme na poslednú spomínanú časť interaktívnej plochy, ktorá je pre túto prácu najzaujímavejšia. Tieto plochy môžeme rozdeliť do piatich základných kategórií, ktoré sú uvedené v následujúcom texte. (Kudale, Wanjale, 2015)

Plochy založené na infračervenom svetle

Takáto plocha sa môže nachádzať kdekoľvek, či už na stene, alebo na stole. Povrch takejto steny však musí byť rovný. Infračervený senzor a vysielač je pripevnený na plochu. Užívateľ svojou interakciou pomocou pera, prsta, alebo iného zariadenia narušuje infračervené svetlo, ktoré sa nachádza tesne nad touto plochou. Pomocou triangulácie je následne vypočítané lokácia bodu interakcie. Hlavnou výhodou tejto technológie je, že je možné použiť akýkoľvek objekt na interakciu (príslušnej veľkosti). Na rozdiel od kapacitných dotykových obrazoviek, obrazovky založené na technológii s infračerveným svetlom nevyžadujú žiadne vzorovanie (napríklad vodivú mriežku), čo v prípade skla zvyšuje odolnosť a optickú priehľadnosť. (Kudale, Wanjale, 2015) (Bhalla M., Bhalla A., 2010)

Na obdobnom princípe funguje aj lokácia pomocou ultrazvuku. Infračervené svetlo má však značné výhody a to rýchlejšiu obnovovaciu frekvenciu a nižšiu latenciu oproti ultrazvuku. Dôvodom je, že svetlo sa šíri oveľa rýchlejšie ako zvuk. Okrem toho umožňuje infračervené svetlo v porovnaní s ultrazvukom pracovať na väčších plochách. (Burdea, Coiffet, 2003)

Odporové plochy

Odporové plochy využívajú membránu, ktorá je umiestnená tesne nad vodivým povrchom. Interakciou užívateľa akýmkoľvek dotykom, či už prstom, alebo nejakým predmetom, sa membrána dotkne vodivého povrchu. Membrána má z vnútornej strany taktiež vodivú vrstvu. Membrána a vodivá plocha pod ňou sú oddelené izolačnými bodmi. Súradnice daného bodu sa jednoducho vypočítajú a vyvolá sa potrebná reakcia. Materiál takejto plochy je veľmi odolný a je ťažké ho poškodiť, citlivý je však na ostré predmety, ktoré ho poškodiť môžu. Tieto plochy sú jednoduché na použitie, avšak pomerne nákladné. Je nutné zdôrazniť, že takáto plocha sa nedá zložiť, vyskytuje sa vždy v jednom kuse, preto je jej transport náročný. Táto technológia sa používala v prvých dotykových zariadeniach, ako mobilné telefóny, multimediálne autorádiá a GPS navigácie. (Kudale, Wanjale, 2015) (Downs, 2005)

Kapacitné plochy

Tento druh interaktívnej plochy pozostáva z vodivej vrstvy drôtikov, typicky vložených za plochu, ktorá má prichádzať do styku s vodivým prvkom. Takýmto prvkom môže byť prst, prípadne takzvaný stylus, ktorý má cievku na jeho špici (je pasívny a nepotrebuje batériu). Na základe vstupu, ktorý zmení elektrický signál na vodivej sieti sa vypočítajú súradnice dotyku. Existujú dva druhy tejto technológie a to povrchové (z anglického *surface capacitive technology*) a projektované (z anglického *projected capacitive technology*) kapacitné plochy. Povrchové kapacitné plochy tvorí iba jedna vodivá vrstva a dotyky sa identifikujú na základe zmeny napätia, súradnice dotyku sa počítajú od rohov displeja. Projektované kapacitné plochy tvoria dve vodivé vrstvy, umiestnené kolmo na seba, ktoré vytvárajú vodivú mriežku (prípadne jedna vrstva, ktorá obsahuje hotovú mirežku). Ovládač dokáže identifikovať zmenu odporu v každom uzle mriežky. Projektované kapacitné plochy teda disponujú vyššou presnosťou a taktiež dokážu identifikovať viacero dotykov súčasne (*multitouch*) oproti povrchovým, ktoré dokážu identifikovať iba jeden dotyk. Kapacitné displeje sa používajú vo väčšine dnešných dotykových zariadení, nevýhodou je vysoká cena. (Kudale, Wanjale, 2015)

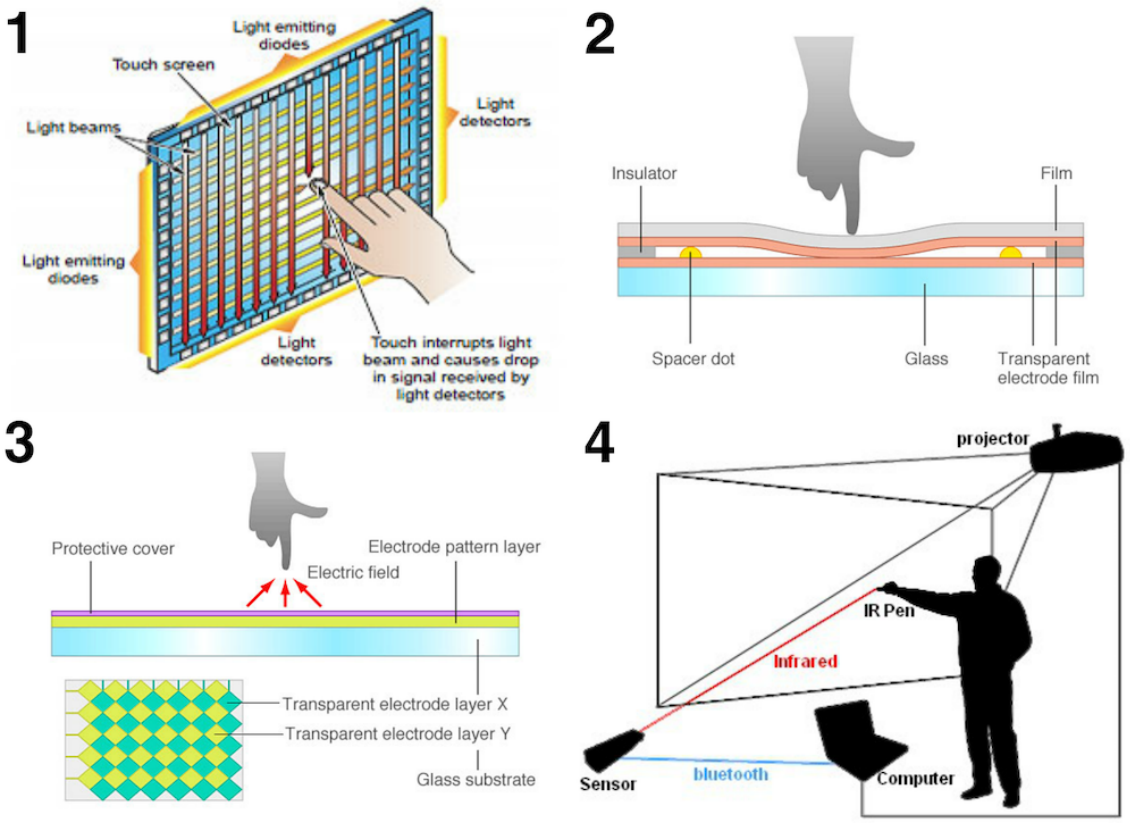
Plochy s Wii ovládačom

Interaktívna plocha je vytvorená pomocou *Wii* ovládača, ktorý je pripojený k počítaču cez Bluetooth. Tento ovládač má infračervenú kameru, ktorá slúži ako senzor, ktorý zachytáva infračervené svetlo vysielané z infračerveného pera. Projektor premieta obraz na plochu a užívateľ používa infračervené pero ako ukazovateľ na interakciu s obsahom. Autorom tohto systému je Johny Lee, ktorý prišiel s týmto prístupom v roku 2007. Tento prístup je užívateľský prívetivý, flexibilný, avšak mierne drahý. (Kudale, Wanjale, 2015) Vrámci využívania *Wii* ovládača je možné použiť aj zabudovaný akcelometer, prípadne ho doplniť prvkom *Motion Plus*, ktorý vylepšuje detekciu komplexných pohybov. Yang and Lee navrhli systém, v ktorom tento ovládač využívali na ovládanie prezentácii a tiež ako bezdrôtovú myš. (Francese, Passero, Tortora, 2012) Tento typ interaktívnej plochy sa v praxi využíva pravdepodobne veľmi málo, narazil som naň iba v rovine projektov, v rámci komerčného využitia som totižto nenašiel spoločnosť, kotrá by takéto riešenie poskytovala.

Interaktívna plocha s interaktívnym projektorom

V takomto systéme je v projektore zabudovaná *CMOS* kamera, ktorá sa správa ako senzor. Projektor premieta obraz na plochu a kamera detekuje pozíciu aktívneho infračerveného pera pri kontakte s plochou, na ktorý je obraz premietaný. Tento systém je veľmi podobný systému s *Wii* ovládačom, pracujú totiž na rovnakom princípe. Podobne ako plocha s *Wii* ovládačom, aj tento typ plochy je v porovnaní s ostatnými systémami pomerne drahý. Taktiež dochádza k niekoľkým problémom

pri ovládaní, napríklad pri strate vizuálneho kontaktu kamery s perom (napríklad pri zakrytí telom). (Kudale, Wanjale, 2015)



Obrázok 1: Ukážky princípu fungovania rôznych typov plôch. 1: Plocha s infračerveným svetlom. 2: Odporová plocha. 3: Kapacitná plocha (projektovaná). 4: Plocha s Wii ovládačom, resp. infračervenou kamerou.
Zdroje: 1: https://goo.gl/s0odoH
2: https://goo.gl/0Atm4m
3: https://goo.gl/6i1yyT
4: https://goo.gl/xiw5bR
Obrázky boli upravené autorom

3.2 Existujúce komerčné riešenia

Na trhu existuje viacero riešení, ktoré sľubujú premenu steny, či stola na interaktívnu. Riešení existuje pomerne veľa, nie každý je však ochotný zdeliť spôsob a technológiu, akou jeho systém funguje. V následujúcej časti si teda rozoberieme tie systémy, ktoré tieto informácie aspoň z časti poskytujú a môžu byť prínosné pre túto prácu.

Ubi

Hlavnými hardwarovými prvkami systému sú *Ubi* kamera (predpokladám so senzorom infračerveného svetla), laserový žiarič a pero s infračerveným svetlom. Túto plochu je možné používať bez akéhokoľvek hardwarového zariadenia a teda ovládať ju iba pomocou prstov, alebo použiť pero, ktoré je súčasťou tohto systému. Dotyky sú detekované pomocou laserového žiariča, ktorý nad plochou vytvorí sieť a je schopný detekovať pozíciu jej narušenia. V prípade, že použijeme na interakciu pero, laserový žiarič nie je potrebný, keďže pero má na špičke zdroj infračerveného svetla, ktorý sa aktivuje po stlačení tejto špičky. Kamera toto infračervené svetlo zachytáva a následne určí jeho pozíciu. Cena takéhoto systému sa pohybuje od necelých 9000 až po niečo vyše 40000 českých korún. Základná cena je za systém, ktorý dovolí interakciu iba ako počítačová myš. Príplatky sú za jednoduché gestá, multitouch gestá a viacerých užívateľov. (Ubi Interactive, 2016)

Obrázok 2: Štandardné zapojenie systému od spoločnosti *Ubi*. V tomto prípade je laserový žiarič umiestnený nad obrazovou plochou.
Zdroj: (Ubi Interactive, 2016)

MotionMagix

Základom tohto systému je buď 3D hĺbková kamera s RGB kamerou, alebo 2D infračervená kamera. Zákazník si môže vybrať tzv. *plug & play* balík, ktorý umožňuje ľahkú inštaláciu a okamžité fungovanie, prípadne si zakúpi iba software tejto spoločnosti a použije vlastný počítač a senzor. Senzor si môže zakúpiť od tejto spoločnosti

za 7500 korún, môže však použiť aj *Microsoft Kinect* senzor. V prípade *plug & play* balíka je súčasťou *MagixBoxTM*, ako tento balík firma sama nazýva, senzor a počítač v jednej krabičke. Tento systém však pracuje iba na vlastnom software a preto je možné používať len u nich zakúpený softwarový obsah (hry, interaktívne plochy). Pri kúpe samotného softwaru je cena 25000 Kč ročne, pri kúpe *MagixBoxTM* balíka cena presahuje hranicu 160000 Kč, software je k tomuto balíku doživotne zdarma. (MotionMagix™, 2015)

Boxlight

Tretí, taktiež pôvodom americký systém od spoločnosti *Boxlight* je možné zakúpiť v rozličných variantoch, ktoré využívajú rozličnú technológiu. Jedná sa buď o aktívne perá s infračerveným svetlom, prípadne o projektor dodávaný s laserovým žiaričom, ktorý je schopný detekovať aj neaktívne prvky na ploche a teda napríklad prsty. Systém s laserovým žiaričom je schopný detekovať taktiež gestá od užívateľa, čo je vrámci systému s aktívnymi perami nemožné. Základný balík *MimioTeach Interactive Whiteboard* je dodávaný bez projektoru a neumožňuje interakciu viacerých užívateľov. (Mimio Interactive Displays, 2017) Cena týchto systémov s projektormi sa pohybuje v rozmedzí od 50 do 150 tisíc českých korún (v závislosti od typu projektoru a poskytnutej výbavy vrámci balíčku ako počet pier, držiakov a podobne). (Mimio Price List, 2016)

Promethean

Táto spoločnosť ponúka rôzne produkty interaktívnych systémov a to interaktívne tabule, interaktívne stoly, interaktívne panely. Spomenuté systémy sú však hardwarovo náročné a majú svojú pevnú veľkosť. Technológie dotyku a rozpoznania gest je teda zahrnutá priamo na dotykovej ploche. Pre túto prácu je zaujímavým produktom interaktívna stena od tejto spoločnosti. Súčasťou tohto produktu je projektor s laserovým žiaričom, ktorý funguje na princípe, ktorý bol vysvetlený vyššie. Systém umožňuje pracovať bez aktívnych pomôcok, rozpoznáva gestá a umožňuje interakciu viacerých užívateľov. Táto stena sa však taktiež ponúka v konrétnych rozmeroch a to 88, 102 a 135 palcov.(Promethean World, 2017) Na webe samotnej spoločnosti sa ceny neuvádzajú, od iných poskytovateľov sa ceny pohybujú v rozmedzí 200 až 400 tisíc českých korún. (Promethean Price List, 2017) (Promethean ActivWall, 2016)

3.3 Existujúce výskumné projekty

Táto sekcia pojednáva o riešeniach, ktoré síce existujú, ale nie sú komerčne využívané. Jedná sa väčšinou o výskumné projekty, ktorých autori zostavili konkrétne riešenie pre nejaký systém, prípadne chceli vyskúšať určitú technológiu a za tým účelom vytvorili systém, ktorý je vlastne dotyková plocha s možnosťou interakcie.

Opäť nebudú opísané všetky riešenia, na ktoré som narazil, vybral som tie, ktoré som považoval za zaujímavé a prínosné pre túto prácu.

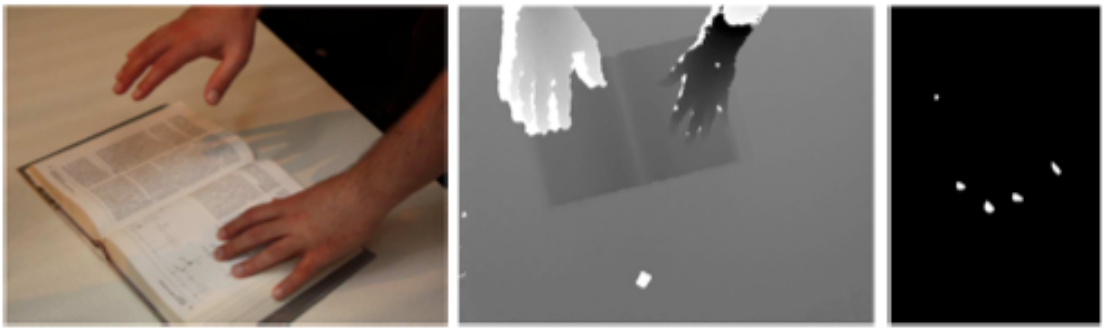
Interaktívna plocha s 3D ovládaním

Tento projekt bol vytvorený v roku 2013. Autori sa pokúsili o vytvorenie interaktívnej plochy s trojdimenzionálnym ovládaním. Základnými prvkami sú interaktívny stôl a hĺbkový senzor umiestný nad týmto stolom. Interaktívny stôl je rozdelený na dve časti, jedna využíva technológiu rozptýleného infračerveného osvetlenia (z anglického *diffused illumination*), druhá časť je bežný LCD displej, ten však pre interakciu zatiaľ nie je funkčný. Interaktívny stôl so spätnou projekciou je teda schopný zachytávať dotyky na ploche pomocou narušenia infračerveného svetla. Tým pádom je možné určiť súradnice dvojrozmerného priestoru. Na určenie tretej dimenzie sa používa práve hĺbkový senzor umiestnený nad stolom. Autori v práci porovnávali dva senzory a to *Kinect* prvej generácie, ako zástupcu lacnejšieho variantu, a *CamCube*, ktorý pracuje na rovnakej technológii ako *Kinect* druhej generácie, ako zástancu vyspelejšej technológie, zároveň 70× drahšej ako *Kinect*. *CamCube* disponoval modernejšou technológiou, avšak poskytoval rozlíšenie iba 204×204 pixelov, naproti tomu *Kinect* 640×480 pixelov. V konečnom dôsledku *Kinect* ohodnotili ako presnejší, najmä kvôli spomínanému rozdielu v rozlíšení. *CamCube* však mal problémy aj s rušením, ktoré spôsobovalo infračervené svetlo na stole, ktoré snímalo dotyky. Trojdimenzionálne ovládanie fungovalo tak, že užívateľ pravou rukou dotykom stola zvolil prvok na obrazovke. Ľavú ruku držal v priestore nad stolom a výškou tejto ruky nad stolom pracoval s vybraným prvkom, napríklad udával množstvo zobrazených infromácií. (Haubner, 2013)

Obrázok 3: Užívateľ uskutočňuje hybridnú interakciu dotykom na objekt pravou rukou a pohybom ľavej ruky smerom hore a dole.
Zdroj: (Haubner, 2013)

Dotykový stôl

Andrew D. Wilson, ktorý je zároveň hlavným výskumníkom a taktiež vedúcim výskumu v spoločnosti *Microsoft*, (Andy Wilson at Microsoft Research, 2017) sa zaoberal projektom na detekciu dotykov na stole pomocou hĺbkového senzora. Ako hĺbkový senzor samozrejme použil *Kinect* prvej genrácie. Senzor bol umiestnený priamo nad stolom a to vo výške 75cm, respektíve 1,5 metra (autor testoval tieto dve vzdialenosti). Práca sa venuje detekcii dotykov aj na nerovnom povrchu (napríklad rôzne objekty na stole). Po každej zmene povrchu je však nutná kalibrácia tohto povrchu, na ktorom bude vykonávaná interakcia. Oblasť dotyku sa vypočítava ako rozdiel hĺbky na jednotlivých pixeloch nových snímok a tých z kalibrácie. Wilson priznáva, že presnosť detekcie dotyku nie je taká vysoká ako pri bežných dotykových systémoch, avšak verí, že presnosť je dostačujúca na využitie pre rôzne aplikácie. Takýto prístup k detekcii dotykov umožňuje zaujímavé výhody, napríklad prácu na nerovnom povrchu, či rôzne bezdotykové interakcie nad týmto povrchom. (Wilson, 2010)

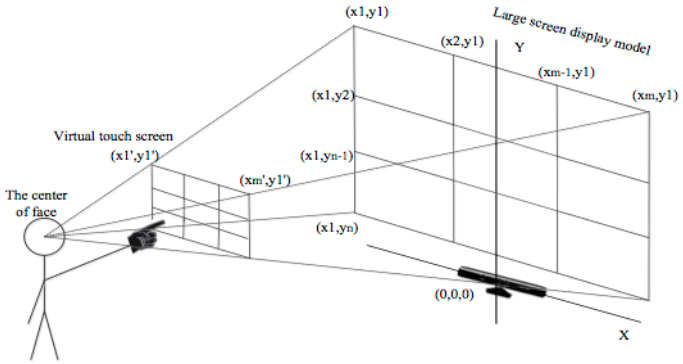


Obrázok 4: Detekcia dotyku na nerovnom povrchu. Vľavo: dotyk knihy. V strede: hĺbkový obraz. Vpravo: zistené dotyky.
Zdroj: (Wilson, 2010)

Virtuálna dotyková plocha

Ako autori uvádzajú, veľa výskumníkov kladie veľkú pozornosť na kostru ľudského tela, takže sa na určovanie smeru, ktorým užívateľ ukazuje, používajú techniky ruky a paže, prípadne ruky a oka. V tomto projekte autori prišli s vlastným riešením, ktoré nepotrebuje určovať smer ukazovania užívateľa. Inšpirovali sa konceptom od Chenga a Takatsuky, ktorí s touto myšlienkou prišli už v roku 2006. Tento koncept mal určité nedostatky, napríklad pevnú vzdialenosť medzi užívateľom a plochou, taktiež dĺžka ruky užívateľa bola určená konštantou, takže tento prístup nevyhovoval pre všetkých užívateľov. Jing a Ye-Peng tento koncept upravili a predstavili upravené riešenie vo vlastnom projekte. Princípom tohto projektu je vytvorenie virtuálnej dotykovej plochy, ktorá je flexibilná a škálovateľná. Virtuálna dotyková plocha sa vytvorí pred prstom užívateľa, ktorým ukazuje na plochu. Veľkosť

a umiestnenie tejto virtuálnej plochy sa vypočíta pomocou 3D koordinácií hlavy a prsta, ktorým užívateľ na plochu ukazuje. Schéma tohto princípu je znázornená na obrázku 5. Na detekciu týchto koordinácií autori používajú *Kinect* prvej generácie. *Kinect* je umiestnený priamo pod plochou, na ktorú užívateľ ukazuje. Virtuálna plocha sa pohybuje vpred a vzad s pohybom prstu, ako aj s pohybom celého tela užvateľa. V tomto projekte bola plocha široká 2,4 a vysoká 1,3 metra rozdelená do šiestich blokov, v každom z nich bola umiestnená žiarovka, ktorá sa pri vyvolaní akcie rozsvietila. Ak užívateľov prst ostane na chvíľu v jednom z blokov virtuálnej plochy, bude vyvolaná akcia v zodpovedajúcom bloku na veľkej ploche. Testovanie bolo vykonávané na šiestich dobrovoľníkoch s rôznou výškou a držaním tela. Autori sa dostali na presnosť 99,2 %, v porovnaní s pôvodným konceptom od Chenga a Takatsuky, ktorý dosahoval presnosť 94,8 %. (Jing, Ye-peng, 2013)



Obrázok 5: Virtuálna dotyková plocha
Zdroj: (Jing, Ye-peng, 2013)

3.4 Kinect

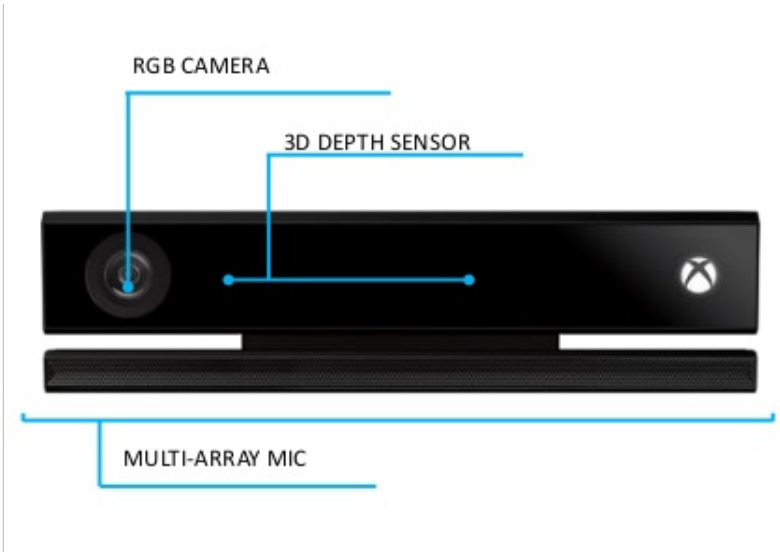
Modelovanie 3D scén, rozpoznávanie gest a sledovanie pohybu sú aktívne a rýchlo sa vyvíjajúce oblasti výskumu. Priemysel s videohrami bol hnaný zvýšeným záujmom o používanie gest, aby bol zážitok z hry ešte zábavnejší a pohlcujúci. Začínajúc nápadom vytvoriť senzor, ktorý umožní hráčom hrať bez držania akéhokoľvek ovládača, spoločnosť *Microsoft* vyvinula *Kinect* senzor. *Kinect* vstúpil na trh 4. novembra 2010 ako doplnok ku konzole *Xbox360*. Vyvinutý bol spoločnosťou *Microsoft* a Izraelskou spoločnosťou *PrimeSense*. *Kinect* vytvoril svetový rekord ako najrýchlejšie predávajúce sa spotrebné zariadenie. Za prvých 60 dní na trhu sa predalo 8 miliónov kusov tohto zariadenia. (Pagliari, Pinto, 2015)

Tento úvod bol venovaný *Kinectu* prvej generácie, na trhu však existuje už aj druhá generácia, ktorú si priblížime a porovnáme s prvou generáciou v následujúcom texte.

Druhá generácia Kinectu

Microsoft Kinect druhej generácie bol sprístupnený v júli 2014. Zariadenie je tvorené dvomi kamerami (klasickou RGB kamerou, infračervenou kamerou) a štyrmi mikrofónmi. Infračervené osvetlenie sledovanej scény je zabezpečené troma infračervenými projektormi. RGB kamera zachytáva obraz v rozlíšení 1920×1080 pixelov, infračervená kamera, ktorá sa pužíva na vytvorenie hĺbkovej mapy, poskytuje rozlíšenie 512×424 pixelov. Výhľadové pole (z anglického *field-of-view*) pre hĺbkové snímanie je 70°horizontálne a 60°vertikálne. Technické špecifikácie udávajú fungujúci rozsah merania od 0,5 až do 4,5 metrovej vzdialenosti od senzoru. Zber týchto údajov je vykonávaný vo frekvencii až 30 snímkov za sekundu. (Lachat, 2015) (Kinect hardware, 2017)

Prvá generácia tohto senzoru poskytovala rozlíšenie RGB kamery 640×512 pixelov a hĺbkovej kamery 640×480 pixelov pri 30 snímkach za sekundu. Výhľadové pole bolo značne menšie a to 58°horizontálne a 45°vertikálne. (Kramer, 2012)



Obrázok 6: *Kinect* senzor druhej generácie
Zdroj: http://ignatiuz.com/blog/kinect-development/kinect-v2-for-windows, upravené autorom

Hĺbkový senzor *Kinectu* druhej generácie funguje na princípe *Time-of-flight* (TOF). Infračervený zdroj vyšle krátke svetelné zhluky, infračervená kamera zachytí ich odraz a vypočíta sa časový rozdiel medzi vyžiarením svetla a jeho zachytením. Na základe týchto údajov dokážeme získať vzdialenosť medzi kamerou a objektom. V tomto smere sa *Kinect* oproti prvej verzii výrazne posunul, keďže hĺbkový senzor prvej generácie fungoval na odlišnom princípe a to kódovaného, respektíve štruktúrovaného svetla. (Amon a kol., 2014) Zdroj infračerveného svetla vyžiari bodkovaný vzor na sledovanú scénu, infračervená kamera zachytí odrazený vzor a vypočíta príslušnú hĺbku pre každý pixel. Hĺbkové meranie je teda vypočítavané pomocou triangulácie. Podstatou je rozdiel, o ktorý je nutné zachytený vzor posunúť, aby sa

zhodoval s vyžiareným vzorom. Hĺbkové dáta prvej verzie *Kinectu* boli slabej kvality, keďže tento prístup nie je dostatočne účinný, aby poskytoval hĺbkovú úplnosť sledovaného obrazu (niektoré hĺbkové body môžu chýbať). (Pagliari, Pinto, 2015)

Kinect prvej generácie dokázal sledovať dve celé postavy pred snímačom a na každej osobe sledoval 20 kĺbov. *Kinect* druhej generácie dokáže sledovať až šesť postáv súčasne a na každej z nich 25 kĺbov. Sledovanie postáv je stabilnejšie, anatomicky presnejšie a ako bolo spomínané, rozšíril sa aj rozsah sledovania. Nové schopnosti infračervenej kamery pomáhajú snímaču vidieť v tme a poskytujú tiež zobrazenie nezávisle na osvetlení. Zabudované štyri mikrofóny zachytávajú a nahrávajú zvuk, zisťujú polohu zdroja zvuku a smer zvukovej vlny. (Kinect hardware, 2017)

Cena *Kinectu* druhej generácie sa pohybuje okolo 2600 Kč. Samotný senzor však nestačí, je potrebné dokúpiť adaptér, ktorý sa predáva samostatne, cena tohto adaptéra sa pohybuje okolo 1300 Kč.

Tabuľka 1: Kinect hardware
Zdroj: (Amon a kol., 2014)(Pagliari, Pinto, 2015)

Komponenta	**Hodnota**
Farebný obraz	1920 × 1080 px
IČ obraz	512 × 424 px
IČ hĺbkový obraz	512 × 424 px
Hĺbkový senzor	0,5–4,5 m
Hĺbková technológia	Time-of-flight
Horizontálny FOV	70°
Vertikálny FOV	60°
Audio	48 kHz, 16-bit
Minimálna latencia	20-60 ms

3.5 Konkurenti Kinectu

Senzor *Kinect* nie je jediný hĺbkový senzor na trhu. Existuje veľké množstvo hĺbkových senzorov, väčšinou však nie sú tak dostupné ako *Kinect*.

Orbbec Persee

Tento senzor je pravdepodobne najnovším konkurentom v tomto sektore. Na trh bol uvedený v decembri 2016. Výrobca udáva, že sa jedná o jediný hĺbkový senzor na trhu s vlastným plnohodnotným počítačom. Znamená to, že môže byť priamo pripojený napríklad k televízoru. Umožňuje prehliadať webové stránky, pozerať videá a hrať hry s jednoduchými hlasovými povelmi a gestami. Systém zatiaľ nepodporuje detekciu celého tela, avšak výrobca sľubuje, že táto funkcia bude čoskoro

doplnená. Vyzerá to, že vývoj takýchto funkcií je iba v začiatkoch a budú postupne doplňované. V tomto smere má senzor *Kinect* veľký náskok. (Orbbec, 2017)

Z hľadiska technických parametrov je na tom tento senzor porovnateľne so senzorom *Kinect*. RGB kamera disponuje nižším rozlíšením a to 1280×720 pixelov pri rýchlosti 30 snímiek za sekundu. Hĺbkový senzor naopak poskytuje o trošku vyššie rozlíšenie 640×480 pixelov. Na určenie hĺbky snímanej scény senzor používa technológiu štruktúrovaného svetla. (Orbbec, 2017)

Cena tohto senzora je približne 6000 českých korún. (Orbbec, 2017)

Intel RealSense R200

Oproti predchádzajúcemu senzoru je tento o čosi starší, konkrétne zo septembra 2015. Tento senzor je veľmi podobný senzoru *Kinect*. Obsahuje tzv. kurzor mód (z angl. *cursor mode*), ktorý ponúka detekciu gest ako krúženie rukami do oboch strán, otvorenie a zatvorenie ruky a gesto pre klik. Rovnako ako *Kinect*, podporuje identifikáciu a sledovanie kostry ľudského tela, avšak iba so šiestimi bodmi na tele. *RealSense R200* disponuje taktiež snímaním a identifikáciou tváre. Tieto údaje sa však týkajú skôr oficiálneho *Intel RealSense* SDK, ako samotného senzora. (Colleen, 2016)

Technické parametre senzora udávajú rozlíšenie RGB kamery 1920×1080 pixelov. Hĺbková kamera funguje na princípe štruktúrovaného svetla a poskytuje obraz s rozlíšením 640×480 pixelov a dokáže detekovať hĺbku obrazu od 0,4 až do 3,5 metra. (Intel ® RealSense™ Camera R200, 2016)

So svojou cenou približne 4200 Kč je tento senzor cenovo pomerne dostupný, avšak momentálne ho nie je možné objednať ani z oficiálnych stránok *Intelu*. (Intel ® RealSense™ Developer Kit (R200), 2017)

ZED

Zaujímavou alternatívou je určite *ZED* od spoločnosti *Stereolabs*. *ZED* pracuje na úplne inej technológii ako všetky spomínané senzory a to stereo videnie.

Systém stereo videnia je spravidla tvorený dvomi RGB kamerami, ktoré sú od seba v určitej vzdialenosti. Táto vzdialenosť zapríčiní to, že každá z týchto kamier zachytáva snímanú scénu pod trochu iným uhlom. Najdôležitešia a časovo najnáročnejšia úloha je registrácia oboch obrázkov a teda identifikácia korešpondujúcich pixelov. Dva pixely sú korešpondujúce, ak reprezentujú rovnaký bod v reálnom svete. Systémy s touto technológiou sa snažia nájsť dvojicu pre každý pixel čoho výsledkom je hustá hĺbková mapa. (Hirschmüller a kol., 2002)

Vďaka odlišnej technológií dokáže *ZED* poskytovať hĺbkový obraz vo vysokom rozlíšení. Na výber sú 4 možné rozlíšenia od 672×376 pixelov pri rýchlosti 100 snímiek za sekundu až po 2208×1242 pixelov pri rýchlosti 15 snímiek za sekundu. Ak chceme rozlíšenie hĺbkoveho obrazu porovnať so senzorom *Kinect*, musíme porovnávať rozlíšenie pri 30 snímkach za sekundu, čo je pri tomto senzore 1920×1080. (ZED - Depth Sensing and Camera Tracking, 2017)

Údaje o rozlíšení hĺbkového obrazu sú priam neuveriteľné, otázna je však ich presnosť. V porovnaní technológie stereo videnia a *time-of-flight* bola úspešnejšia druhá z nich. *Time-of-flight* si počínalo lepšie ako aj v presnosti, tak aj v rýchlosti. Nevýhodou tejto technológie je nižšie rozlíšenie a možné problémy pri určitých svetelných podmienkach. (Kazmi, 2014)

Senzor *ZED* je však v úplne inej cenovej kategórii ako jeho konkurenti. Na stránkach výrobcu je možné tento senzor objednať približne za 11000 Kč. (ZED - Depth Sensing and Camera Tracking, 2017)

3.6 Frameworky pre prácu s Kinectom

Pre prácu s Kinectom a teda pre komunikáciu s ním existuje viacero frameworkov. V tejto sekcii je predstavených niekoľko z nich, zároveň sú zhodnotené ich plusy a mínusy pre prácu s nimi.

Kinect for Windows SDK 2.0

Jedná sa o oficiálny framework od spoločnosti *Microsoft* vydaný v Októbri 2014. Podporovaný operačný systém pre tento framework je *Microsoft Windows 8* a vyšší, podporované programovacie jazyky pre tento framework sú C++, C#, Visual Basic alebo akýkoľvek iný jazyk platformy .NET. Táto sada zahrňuje vzorové aplikácie s prístupom k úplnému vzorovému kódu, tiež program *Kinect Studio* a prostriedky pre zjednodušnie a zrýchlenie vývoja aplikácií. Za užitočný doplnok považujem *NuGet*, čo je vlastne správca balíkov pre *Visual Studio*. Spomínaný program *Kinect Studio* dokáže prehrávať aktuálne snímky z *Kinectu*, nahrávať ich a neskôr si ich znova prehrať. Samotný framework dokáže poskytovať údaje až o šiestich osobách, na každej sleduje 25 kĺbov. *Microsoft* na svojich stránkach uvádza anatomicky presnejšie pozície tela s rýchlou interakciou, presnejších avatarov a taktiež vylepšené sledovanie tváre. Uvedená je aj vylepšená podpora simultánneho využitia vo viacerých aplikáciách, ktorá umožňuje, aby jeden snímač súčasne používalo niekoľko aplikácií. (Kinect tools and resources, 2017)

Výhodami tejto sady sú doplnkové prostriedky a fakt, že sa jedná o oficiálny framework, takže je možné počítať s neustálou podporou a jeho ďalším vývojom. Nevýhodou tohto balíka je slabá dokumentácia (ak porovnám dokumentáciu s inými dokumentáciami, s ktorými mám skúsenosť).

Libfreenect2

Libfreenect2 je open source multiplatformný ovládač pre *Kinect* druhej generácie od komunity *OpenKinect*. Tento ovládač umožňuje spracovanie farebného, infračerveného a hĺbkového obrazu, obsahuje viacero implementácií GPU a hardwarových zrýchlení pre spracovanie týchto obrazov. *Libfreenect2* funguje samostatne a nevyžaduje oficiálny framework pre jeho fungovanie. Oproti oficiálnemu frameworku

však obsahuje iba najzákladnejšie funkcie a teda prenos jednotlivých obrazov zo zariadenia. (Libfreenect2: API Reference, 2016)

Výhodou je, že je multiplatformný a teda je možné ho použiť pod akýmkoľvek systémom a s väčším výberom programovacieho jazyka ako pri originálnom frameworku. Nevýhodou je, že sa jedná o open source ovládač a teda nie je zabezpečná stabilita, podpora do budúcna a otázna je aj kompletnosť dokumentácie. Ako nevýhodu by som uviedol aj inštaláciu tohto ovládača, ktorá je z dôvodu multiplatformnosti pomerne zložitá.

Vitruvius

Tento framework je postavený na oficiálnom frameworku od spoločnosti *Microsoft* a tvorí akúsi jeho nadstavbu. Zjednodušuje vývoj spojený s *Kinectom* vo veľa aspektoch. Pre príklad, uľahčuje proces ovládania avataru vlastným telom a to pomocou jedného riadku kódu. Obsahuje veľké množstvo ťažkej matematiky, ktorá súvisí so sledovaním pohybu tela, počíta uhly v kĺboch, výšku užívateľa, dĺžku špecifikovaných segmentov v 3D priestore, rotáciu tela vo všetkých troch osách. Obsahuje zjednodušenú manipuláciu s bitmapami, detekciu gest a tváre. Jedná sa o veľký framework, ktorý podporil aj sám *Microsoft* na svojom blogu.

Tento framework sa ponúka v štyroch rôznych variantoch. Prvý je zdarma a obsahuje iba minimum pridaných funkcií, slúži najmä na vyskúšanie tohto frameworku. Ďalšie tri varianty poskytujú viac a viac odomknutých funkcií, aktualizácie zdarma, podporu na telefóne, či 24 hodinovú odozvu na podporu. Najlacnejší balík, ak nepočítame balík zdarma, stojí približne 2500 korún, nasleduje ho balík za 5000 a najdrahší balík je možné získať za cca 22500 korún. (Vitruvius) (Vitruvius simplifies development of Kinect for Windows apps, 2017)

Výhodu tohto frameworku vidím v jednoduchosti a uľahčení bežných úkonov pri práci s Kinectom. Propagácia tohto produktu samotným microsoftom vo mne vzbudzuje určitú dôveru pre tento framework. Nevýhodou je cena pri potrebe použiť zložitejšie funkcie, ktoré sú za príplatok.

Zhrnutie

V nasledujúcej tabuľke sú zhrnuté podstatné informácie týkajúce sa predstavených frameworkov.

Tabuľka 2: Porovnanie frameworkov pre prácu s *Kinectom* (dokumentácia je hodnotená 1–5, kde 1 je najlepšie).
Zdroj: viď. zdroje k jednotlivým podsekciám

Vlastnosť	**Kinect for Windows SDK 2.0**	**Libfreenect2**	**Vitruvius**
Funkcionalita	dáta, skeleton, tvár, avatar	dáta	dáta, skeleton, tvár, avatar, gestá
Dokumentácia	3	3	1
Posledná aktualizácia	21.10.2014	pravidelne aktualizované	1.8.2016
Cena	0 Kč	0 Kč	0–22500 Kč
Platforma	Windows	Windows, Linux, macOs	Windows
Komentár	oficiálny framework, uzavretý	open-source	zjednodušená syntax, uzavretý

Komentár v tabuľke *uzavretý* znamená, že daný framework nie je možné upraviť. Naproti tomu stojí *open-source* framework, ktorý verejne poskytuje zdrojový kód a programátor si ho môže upraviť podľa vlastných potrieb.

4 Metodika

Na základe vykonanej rešerše budú vykonané následujúce kroky pre vytvorenie vlastného riešenia problematiky, ktorou sa táto práca zaoberá.

Ako z teoretickej časti vyplýva, väčšina podobných projektov využíva na detekciu dotykov a gest infračervené svetlo (v zmysle infračerveného žiariča a prijímača), prípadne hĺbkovú kameru. Vrámci tejto práce som sa rozhodol použiť hĺbkový obraz na detekciu dotykov a gest a senzor *Kinect* druhej generácie ako zdroj tohto hĺbkového obrazu.

Čo sa týka umiestnenia senzoru, plánujem otestovať dva varianty a na základe ich výsledkov sa rozhodnúť pre konečné umiestnenie. Plánované umiestnenie obrazovej plochy pre obe zapojenia je na stene (stena zviera s podlahou pravý uhol). Konkrétne ide o snímanie hĺbky obrazu cez polopriehľadnú stenu (senzor bude umiestnený za touto stenou, čiže užívateľ ho neuvidí), druhý variant je umiestniť senzor nad, resp. pod obrazovú plochu (senzor a stena zviera pravý uhol). Tieto varianty boli vybrané na základe rešerše, kde sa takéto umiestnenia senzoru neobjavili a preto považujem za zaujímavé ich vyskúšať. Lokáciu dotyku plánujem určiť pomocou údaja o zmene hĺbky v určitom bode a pozície tohto bodu na ose x a y v prijatej snímke.

Výber vhodného frameworku pre túto prácu nie je kľúčový. Z hľadiska umiestnenia *Kinectu* je nemožné používať dáta o užívateľovi priamo z *Kinectu*, pretože užívateľ sa nebude nachádzať v snímanom obraze v podmienkach, ktoré si *Kinect* vyžaduje (*Kinect* nebude snímať celého užívateľa, ale iba jeho ruky, prípadne vrchnú časť tela, čo na detekciu tela a gest nestačí). Výber síce môže pozitívne, prípadne negatívne ovplyvniť úvodnú rýchlosť vývoja, avšak v ďalšej fázi nie. Dôvodom je, že táto práca bude využívať iba "surové" dáta zo senzoru, žiadnu inú jeho funkcionalitu (trackovanie tela a podobne). Na základe vykonanej rešerše sa teda prikláňam k oficiálnemu frameworku od *Microsoftu* pre *Kinect* druhej generácie, prípadne jeho nadstavby *Vitruvius* od spoločnosti *LightBuzz*.

Ako z predchádzajúceho odstavca vyplýva, ďalším krokom bude navrhnúť a implementovať vlastný systém detekcie dotyku, prípadne zvolených gest. Čiastkové kroky tejto časti závisia od výsledku testovania umiestnenia senzoru.

Zvoleným programovacím jazykom je C#. Dôvody sú jednoduché, je to oficiálne podporovaný jazyk pre prácu s *Kinectom* a preto v tomto smere očakávam najmenej problémov. Spoločnosť *Microsoft* má tiež vlastné vývojové prostredie *Visual Studio* (konkrétna použitá verzia bude 2015), ktoré štandardne C# podporuje.

5 Vlastná práca

Táto kapitola pojednáva o procese tvorby vlastného riešenia dotykovej plochy s použitím hĺbkového senzora.

5.1 Umiestnenie senzora

V metodike práce je uvedené vyskúšanie dvoch rôznych variantov umiestnení senzora *Kinect* a vybrať vhodnejší z nich pre ďalšiu prácu. Jedno z umiestnení si vyžaduje polopriehľadnú stenu a spätnú projekciu. Tento fakt znamená priestorovú náročnosť, ako aj nutnosť technickej vybavenosti. Druhé z umiestnení nie je náročné na priestor, ako ani na technickú vybavenosť. Vyžaduje si iba obyčajnú stenu, senzor *Kinect* a akýkoľvek projektor. Projektory s krátkou projekčnou vzdialenosťou sú však oveľa vhodnejšie a to z dôvodu vytvárania menších tieňov na premietaný obraz a teda užívateľ si netieni na tak veľkú časť obrazovky.

Za polopriehľadnou stenou

V prvom variante je senzor Kinect umiestnený za polopriehľadnou stenou a interakciu užívateľa sníma cez túto stenu. Užívateľ teda senzor nevidí a senzor nijako neobmedzuje jeho pohyb. Vzdialenosť senzora od steny bola približne dva metre, čo postačovalo na snímanie celej plochy steny. Pri tomto variante je nutné myslieť na to, že za polopriehľadnou stenou musí byť dostatočný priestor na umiestnenie senzoru, najmä horizontálna vzdialenosť od tejto steny, potrebná pre snímanie požadovanej plochy.

Obrázok 7: Snímky scény pri umiestnení senzora *Kinect* za polopriehľadnou stenou. Vľavo: RGB kamera. Vpravo: Hĺbkový obraz.

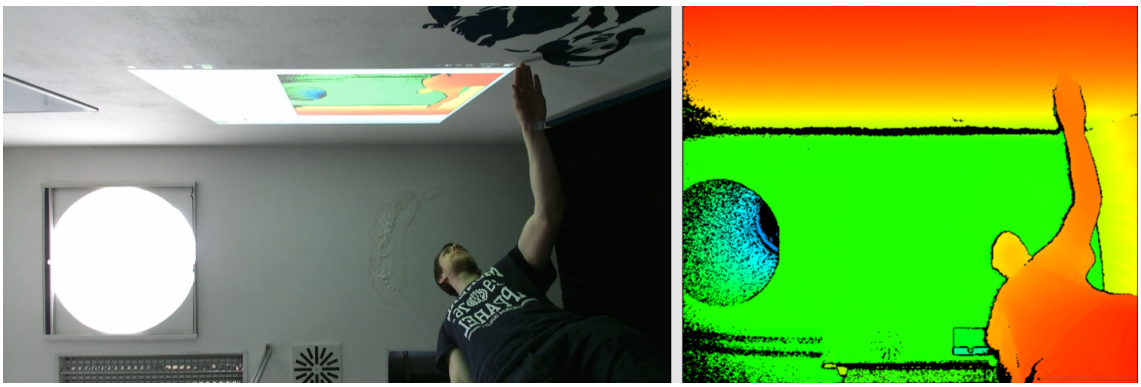
Ako vidíme na obrázku 7, hĺbkový senzor zachytáva iba minimálne rozdiely v hĺbke steny a užívateľom za stenou. Na snímke z RGB kamery je jasne vidieť užívateľa, hĺbková kamera však tieto údaje na prvý pohľad nezachytáva. Navyše, na okrajoch snímaného obrazu dochádza k veľkému šumu, čiernou farbou sú označené nevalidné pixely, senzor Kinect nedokázal určiť ich hĺbku.

Pod stenou

Druhý variant využíva obyčajnú stenu v akejkoľvek miestnosti. Podmienkou je, aby bola stena rovná (bez výstupkov a zakrivení) a aby s podlahou zvierala pravý uhol. Senzor *Kinect* je umiestnený na podlahe (najmenej 30 cm od steny) a so stenou taktiež zviera pravý uhol. Výškové umiestnenie obrazovej plochy súvisí s možnou fyzickou šírkou obrazu. Horizontálne FOV senzora *Kinect* je 70°. Od toho sa odvíja možná fyzická šírka premietaného obrazu vo vzťahu k vertikálnej vzdialenosti od senzora. To znamená, že možná fyzická šírka obrazu je priamo úmerná vertikálnej vzdialenosti obrazovej plochy od senzora *Kinect*. Pozor si však treba dať na minimálnu vertikálnu vzdialenosť 50 centimetrov a maximálnu možnú vzdialenosť od senzora *Kinect*, ktorá je 4,5 metra.

Obrázok 8: Snímky scény pri umiestnení senzora *Kinect* pod polopriehľadnou stenou. Vľavo: RGB kamera. Vpravo: Hĺbkový obraz.

Druhý variant bol najprv zostavený s polopriehľadnou stenou, aby bolo prípadne možné využívať spätnú projekciu pre lepšiu ovládateľnosť, keďže by si užívateľ netienil na premietaný obraz. Na obrázku 8 však opäť môžeme vidieť veľký šum v oblasti steny a detekcia dotykov by tak bola príliš náročná.



Obrázok 9: Snímky scény pri umiestnení senzora *Kinect* pod obyčajnou stenou. Vľavo: RGB kamera. Vpravo: Hĺbkový obraz.

Umiestnenie senzora *Kinect* pod obyčajnou stenou už na prvý pohľad poskytuje najpresnejšie údaje. K určitému šumu dochádza v rohu medzi stenou a stropom, to by však pri detekcii dotyku nemalo prekážať. Ďalší šum, ktorý je možné na obrázku 9 pozorovať spôsobujú svetlo odrážajúce predmety (*Kinect* používa na detekciu hĺbky infračervené svetlo, ktoré však tieto objekty odrazia a senzor teda nie je schopný určiť ich vzdialenosť od senzora).

Porovnanie

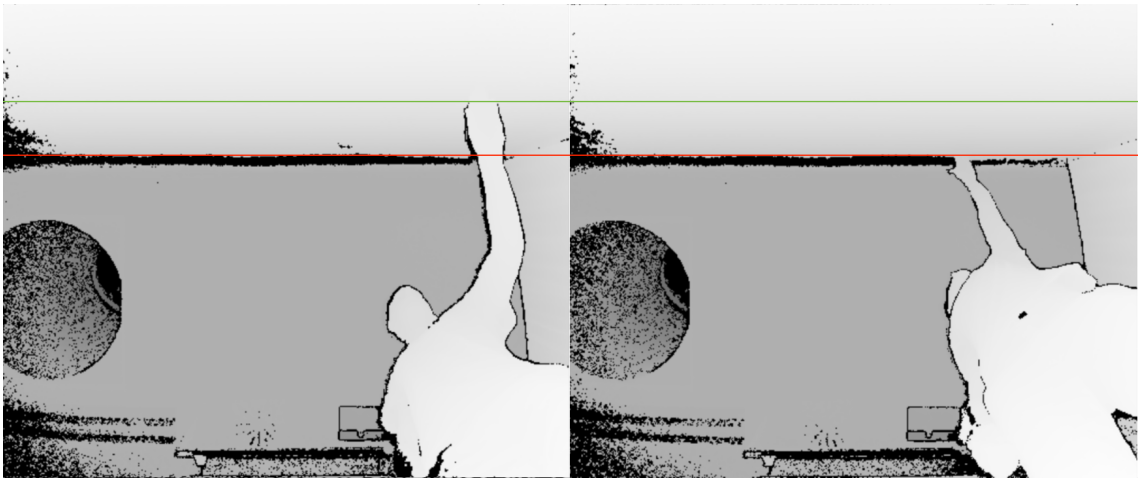
Pre každý z týchto troch variantov, ktoré boli predstavené bolo vykonané testovanie za účelom výberu najvhodnejšieho z nich. Pre testovacie účely bol implementovaný jednoduchý program, ktorý dokázal zafarbovať každý pixel farbou, ktorá odpovedá jeho hĺbke. Takúto funkcionalitu ponúka aj program *Kinect Studio*, z ktorého sú vytvorené predchádzajúce obrázky. Vlastný implementovaný program však navyše ponúka možnosť stanoviť si hĺbkovú hranicu a za touto hranicou zafarbovať pixely odlišnou farbou, než pixely, ktoré sa nachádzajú pred ňou. Cieľom bola možnosť viditeľnejšie sledovať snímanú hĺbku *Kinectu*. Ukázalo sa, že všetky tri umiestnenia dokázali zachytiť hĺbku obrazu pomerne dobre. Najväčší šum a s ním spojené problémy sa objavili pri prvom zapojení. *Kinect* síce dokázal odmerať hĺbku obrazu aj za polopriehľadnou stenou (avšak iba do určitej vzdialenosti), ale s veľkou nepresnosťou, objekty mali okolo seba príliš veľký šum, ktorý by detekciu dotykov skresľoval. Problémom bol aj pohyb plátna samotnej polopriehľadnej steny, ktoré sa aj pri minimálnom dotyku začínalo vlniť, čo spôsobovalo výkyvy v nameranej hĺbke. Podobné problémy s polopriehľadnou stenou sprevádzali aj druhý variant umiestnenia senzoru a to pod touto stenou. Opäť dochádzalo k príliš veľkému šumu a nepresnostiam. Najlepší výsledok bol dosiahnutý pri umiestnení senzora k obyčajnej stene. Nevýhodou tohto umiestnenia je nutnosť použitia obyčajného projektoru a teda užívateľ si bude vytvárať na premietaný obraz tieň. Napriek tomu práca pokračuje práve s týmto umiestnením, pretože vykazoval najväčšiu presnosť merania.

5.2 Návrh riešenia

Senzor *Kinect* posiela údaje o hĺbke 30 krát za sekundu s rozlíšením obrazu 512×424 pixelov. Ak si teda spočítame počet pixelov v jednej snímke, dôjdeme k číslu 217088. Toto je však iba jeden obraz, takýchto obrazov príde za sekundu 30. Za sekundu teda zo senzoru Kinect príde cez 6,5 milióna hodnôt iba o hĺbke obrazu. V prípade zobrazovania, či analýzy celého obrazu je takýto úkon výpočtovo pomerne náročný. Na redukciu výpočtovej náročnosti sa môže použiť napríklad Gaussova pyramída, ktorú aj v tejto súvislosti (a na kompenzáciu skreslenia údajov) použil Haubner a kolektív. (2013) Riešenie tejto práce sa však uberalo iným smerom a to snahe znížiť počet pixelov, ktoré je nutné analyzovať, na minimum. Prvotný návrh teda pojednáva o akejsi hranici, ktorá by mala na starosti detekciu začiatku a ukončenia dotyku.

Postup pri tvorbe návrhu

Ako už bolo naznačené, prvým návrhom bola jednoduchá myšlienka pomyselnej hranice, ktorú keď užívateľ presiahne, aktivuje dotyk. Pri inicializácii systému by si táto hranica uložila prvotné hodnoty hĺbky jednotlivých pixelov. Následne by tieto uložené hodnoty porovnávala s novými hodnotami. Pri určitej zmene by bolo detekované narušenie hranice a teda začiatok nového dotyku. Na prvý pohľad jednoduchý a účinný spôsob detekcie dotyku. Pri hlbšej analýze sa však objavujú skryté problémy.



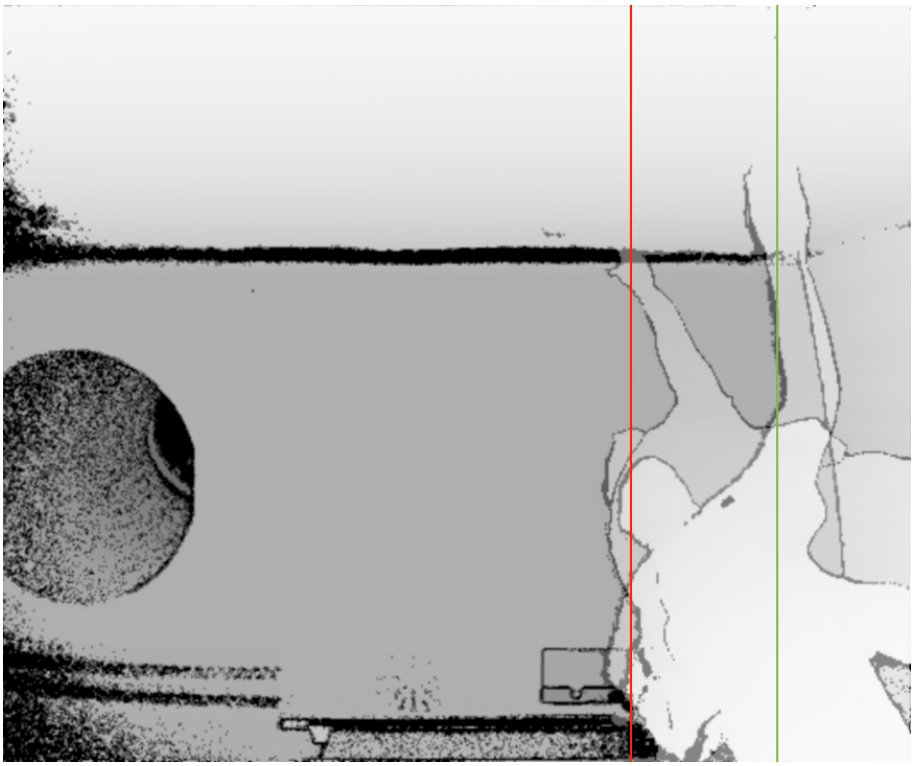
Obrázok 10: Stanovenie jedinej hranice na snímanie dotyku. Červená hranica je v hornej časti premietaného obrazu, zelená hranica v spodnej časti.

Na obrázku 10 ihneď vidíme prekážku, prečo systém nemôže fungovať na takomto jednoduchom princípe. Stanovenie jedinej hranice je totižto prakticky nemožné. Ak si túto hranicu určíme príliš nízko (zelená hranica), nedokáže nám zachytávať dotyky vo vrchnej časti obrazovky. Ak si ju určíme príliš vysoko (červená hranica), dokáže zachytávať dotyky na celej ploche obrazovky, avšak dotyky v dolnej časti by boli iniciované príliš skoro. Tento jav spôsobuje FOV kamery, ktorá obraz sníma. Čím je vzdialenosť od senzora väčšia, tým širší obraz kamera zachytáva, avšak stále na rovnakom počte pixelov, preto dochádza k tomuto posunu.

Návrh sa teda rozšíril na viacero takýchto hraníc. Detekcia dotykov na celej oblasti obrazovky je teda vyriešená. Problém s detekciou príliš včasného dotyku v dolnej časti obrazovky však stále pretrváva. Preto sa každá z hraníc bude starať iba o určitú časť obrazovky, konkrétne po nižšiu hranicu.

Rovnaký problém nastáva aj pri sledovaní šírky obrazu, čo je možné vidieť na obrázku 11. V dolnej časti šírka premietaného obrazu zaberá takmer celú šírku obrazu poskytnutým senzorom Kinect. Ak je vykonaný dotyk v ľavom dolnom rohu obrazovky, súradnica na ose x v jednotkách pixelov je napríklad 445. Znamená to teda, že ruka sa na obraze zo senzora *Kinect* nachádza 445 pixelov po ose x od ľavého okraja. Dotyk v hornej ľavej časti však nebude mať rovnakú hodnotu, čo je

spôsobené, ako aj v predchádzajúcom prípade, FOV kamery, ktorá obraz sníma. Pre predstavu, v prípade reálnej výšky premietaného obrazu približne 90 centimetrov je ľavý horný roh obrazovky na ose x umiestnený 365 pixelov od okraja. Znamená to teda, že rozdiel medzi ľavým spodným a horným bodom obrazovky na ose x je 80 pixelov. Zohľadnenie tejto skutočnosti pri implementácii je nevyhnutnou samozrejmosťou.



Obrázok 11: Porovnanie súradníc ruky na ose x pri dotyku v hornej a dolnej časti premietaného obrazu. Červenou čiarou je vyznačený dotyk v hornej časti premietaného obrazu, zelenou v spodnej časti (obrázok sa skladá z dvoch rôznych snímiek).

Finálny návrh

Na základe vyššie spomenutých faktov, ktoré boli zistené analýzou návrhu riešenia, či skutočným testovaním došlo, k tomuto finálnemu návrhu. Je nutné zdôrazniť, že v tejto podkapitole je návrh systému opísaný veľmi stručne, detaily ohľadom konkrétnej funkčnosti sú opísané v implementačnej časti.

Detekcia dotyku bude vykonávaná pomocou niekoľkých hraníc, ktoré budú kontrolovať vstup cudzieho objektu do nimi snímanej scény. Každá z hraníc si uloží prvotné hodnoty, ktoré budú následne porovnávané s tými novými. Pri dosiahnutí určitého rozdielu (je možné ho ľubovoľne nastaviť) na niektorej z hraníc sa inicializuje nový dotyk. Ukončenie narušenia hranice znamená ukončenie dotyku. Každá hranica kontroluje iba určitý úsek na stene a to ako do šírky, tak aj do hĺbky.

Hodnoty týchto hraníc budú vypočítané pri inicializácii tohto systému. Inicializácia bude prebiehať uložením prvotných hodnôt a určením pozície rohov obrazovky. Rohy obrazovky sa určia dotykom od užívateľa do každého z nich. Tento návrh by mal byť dostačujúci pre fungovanie systému a tvorbu implementácie.

Otestovať tento systém bude možné na vlastnej aplikácii s tématikou pre sociálnu sieť *Twitter*. Obrazovka bude rozdelená na toľko pracovných plôch, koľko si užívateľ zvolí. Na každej pracovnej ploche uvidí užívateľ tweety zo svojej domovskej stránky na *Twitteri*. V spodnej časti obrazovky bude spoločná časť pre všetkých užívateľov. Táto časť obrazovky bude slúžiť na retweet ľubovoľného tweetu. Retweet prebehne z účtu užívateľa, z ktorého pracovnej plochy bol tweet pretiahnutý do tejto oblasti. Užívatelia si taktiež môžu tweety vymieňať medzi sebou. Užívateľ teda môže retweetnuť tweet zo susednej pracovnej plochy (tweet z domovskej stránky iného užívateľa), avšak najprv si ho bude musieť pretiahnúť na svoju plochu a následne do retweet zóny.

5.3 Implementácia systému na detekciu dotykov

V nasledujúcej časti je priblížená implementácia a podrobný opis fungovania systému a testovacej aplikácie.

Práca so senzorom Kinect

Pre prácu s *Kinectom* bol použitý oficiálny framework *Microsoft Kinect for Windows SDK 2.0*. Dôvodom bolo, že so samotným *Kinectom* sa pracuje iba minimálne. *Kinect* pre systém dodáva hĺbkový obraz scény, všetko ostatné je však riešené už iba s týmito dátami, bez ohľadu na to, odkiaľ pochádzajú, či už zo senzora *Kinect*, alebo akéhokoľvek iného hĺbkového senzora.

Prvým krokom je inicializácia senzora *Kinect* a registrácia metódy, ktorá bude volaná pri určitej udalosti, konkrétne pri príchode novej snímky zo senzora *Kinect*.

```
KinectSensor sensor;
MultiSourceFrameReader reader;
...

this.sensor = KinectSensor.GetDefault();

if (this.sensor != null)
{
   this.sensor.Open();

   this.reader =
       this.sensor.OpenMultiSourceFrameReader(FrameSourceTypes.Color |
       FrameSourceTypes.Depth | FrameSourceTypes.Infrared);
   this.reader.MultiSourceFrameArrived += Reader_MultiSourceFrameArrived;
}
```

Samotné fungovanie systému nevyžaduje potrebu infračerveného a farebného obrazu, avšak pre kalibračné a ukážkové účely sú zaregistrované aj tieto zdrojové typy. V metóde, ktorá bola registrovaná na udalosť prijatia novej snímky, budú vykonané potrebné úkony. V zásade sa jedná o jedno volanie metódy, bez ohľadu na typ snímky (hĺbková, farebná, infračervená), ktorá prišla.

```
camera.Source = frame.ToBitmap();
```

Premenná *camera* odkazuje na vizuálny prvok, kontrétne *ImageViewer*. Metóda *ToBitmap* je preťažená metóda, ktorá má tri rôzne implementácie v závislosti od predaného parametru. Volaním metódy sa žiaden parameter nepredáva, jedná sa totižto o rozšírenie triedy *ImageSource* o túto metódu, kde samotným parametrom je objekt, ktorý túto metódu volá, v tomto prípade *frame*.

Prijaté dáta sa spracúvajú podľa potreby. Dáta z farebnej a infračervenej kamery sa jednoduchým spôsobom prevedú na obrázok, ktorý je potom možné zobraziť na obrazovke. Pre túto prácu nie sú detaily ohľadom tejto implementácie dôležité, preto bude podrobnejšie opísaný iba proces spracovania hĺbkového obrazu.

Pre spracovanie hĺbkového obrazu sa teda v metóde *ToBitmap* dáta z objektu typu *DepthFrame* prekopírujú do jednoduchého poľa, ktorý obsahuje dátový typ *ushort*. Dáta reprezentujú hĺbku snímky v jednotlivých pixeloch. Znamená to, že spomínané pole alokujeme o veľkosti násobku šírky a výšky obrazu. Objekt *frame* obsahuje množstvo rozličných atribútov a metód, ktoré však pre účely detekcie dotyku nie sú potrebné, preto je zbytočné uchovávať si celý tento objekt, pre ďalšie spracovanie postačuje pole hĺbkových hodnôt.

V prípade prevodu prijatých dát na obrázok je vytvorené ďalšie pole, ktoré obsahuje 4× viac hodnôt ako pole s hĺbkovými údajmi. V tomto poli sa nachádza reprezentácia obrázka, kde každý pixel reprezentujú 4 bajty (červená, zelená, modrá farba a alfa kanál). Prevod hĺbky pixelu na farbu záleží od žiadaného výsledku. Môže sa jednať o farebný, či čiernobiely obrázok, obrázok iba v dvoch farbách (rozdelenie podľa určitej hĺbky) a podobne.

```
int mapDepthToByte = maxDepth / 256;

byte intensity = (byte)(depth >= minDepth && depth <= maxDepth ? (depth /
    mapDepthToByte) : 0);
```

Pre čiernobiely obrázok si jednoducho povedané vypočítame svetlosť pixelu. Premenná *depth* disponuje hĺbkovým údajom v mm pre aktuálny pixel (ukážkový výpočet sa nachádza vo *for* cykle). Hodnota *intensity* sa nastaví pre všetky farby (červenú, zelenú, modrú) reprezentujúce daný pixel, hodnota alfa kanálu je nastavená na 255. Tento jednoduchý výpočet umožní zobraziť čiernobiely obrázok, kde sú tmavšie pixely bližšie ku senzoru *Kinect*, ako svetlejšie. Čím je teda pixel od senzora *Kinect* ďalej, tým je svetlejší. Čierny pixel znamená najbližšiu možnú hĺbku, prípadne nevalidný pixel, keďže nadobúda hodnotu 0 pre všetky tri reprezentujúce farby. Pozor, predchádzajúce obrázky v texte nie sú zhotovené z vlastného programu, ale

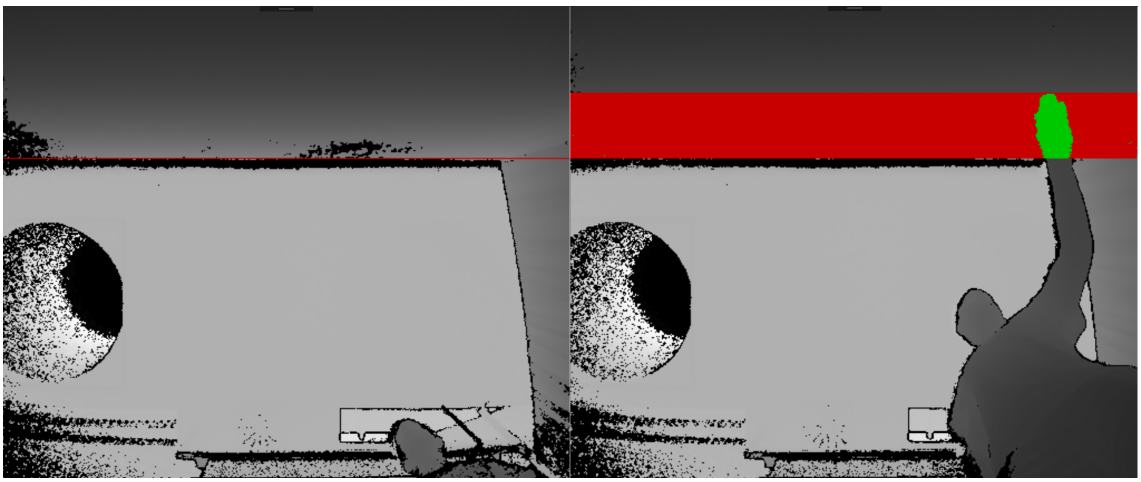
ako je spomínané, z programu *Kinect Studio*, ktorý používa opačné zafarbenie pixelov.

V tomto bode sa končí priama práca so senzorom *Kinect*. Do systému na detekciu gest sa pošle iba vytvorené pole, ktoré obsahuje hodnoty hĺbky v jednotlivých pixeloch, žiadne iné údaje k ďalšiemu spracovaniu nie je potreba. Samotný systém na detekciu dotykov teda nie je nevyhnutne viazaný na senzor *Kinect*, môže sa jednať o akýkoľvek iný senzor, ktorý dokáže poskytovať dáta o hĺbke snímanej scény pre jednotlivé pixely.

Inicializácia systému na detekciu dotykov

Ako už bolo v predchádzajúcom texte naznačené, systém bude fungovať na princípe hraníc, ktoré budú detekovať narušenie ich snímaného priestoru. Porovnávať budú nové hodnoty, ktoré prichádzajú zo senzora *Kinect* s tými, ktoré prišli ako prvé a sú uložené pre tento účel. Pri spustení si teda systém uloží dáta z niekoľkých snímiek, počet je možné meniť. V tomto prípade sa pracovalo s hodnotou 50 snímiek, teda inicializácia systému trvá približne 2 sekundy (keďže senzor *Kinect* posiela 30 snímiek za sekundu). Tieto hodnoty sa ukladajú do pripraveného poľa. Druhý prijatý snímok sa neukladá ako celok, ale kontrolujú sa iba chybné pixely. Ak sa v prvom snímku nachádzal nevalidný pixel, skontroluje sa hodnota tohto pixelu v novom snímku. V prípade, že nový snímok obsahuje validnú hodnotu hĺbky, uloží sa nová hodnota. Takto je dosiahnutý pomerne presný obraz hĺbky snímanej scény v každom bode.

Následne sa od užívateľa očakáva kalibrácia plochy, ktorú je potrebné kontrolovať pre prípadné dotyky. Táto kalibrácia prebieha dotykom do každého zo štyroch rohov premietaného obrazu, čím sa určí veľkosť a poloha tohto obrazu. Systém zatiaľ disponuje iba jedinou hranicou dotyku, ktorá je umiestnená tak, aby pokrývala celú stenu. Problém včasného dotyku opísaného pri návrhu riešenia v tomto prípade nie je prekážkou, nejedná sa o samotnú interakciu, iba o kalibráciu. V prípade detekcie dotyku na danej hranici sa kontroluje hranica o pixel vyššie. Tento proces sa opakuje, až kým sa nenarazí na hranicu bez detekcie dotyku. Takto sa zistí posledná hranica, na ktorej je dotyk detekovaný, čo bude neskôr použité pre vytvorenie všetkých hraníc. Presný popis detekcie dotyku je opísaný v ďalšom texte, táto podsekcia sa zameriava iba na inicializáciu systému.



Obrázok 12: Inicializácia systému. Vľavo: Jediná hranica dotyku, ktorá sníma celú stenu. Vpravo: Dotyk v spodnej časti obrazovky.

Pre lepšiu predstavivosť poslúži obrázok 12, na ktorom vidíme vyššie popísaný postup. Pri dotyku sa prehľadá ďalšia hranica na snímke, či obsahuje dotyk, táto hranica je vykreslená červenou farbou. Pixely, v ktorých je detekovaná zmena hĺbky sú vyznačené zelenou farbou. V pravej časti obrázka je možné vidieť poslednú nájdenú hranicu, v ktorej sa už dotyk nenachádza. Táto hranica, ako aj nameraná hĺbka dotyku v tomto bode, je uložená a bude použitá pri výpočte umiestnenia zobrazovanej plochy.

Po uskutočnení dotykov do všetkých štyroch rohov (v ľubovoľnom poradí) systém identifikuje, ktorý dotyk patrí do ktorého rohu. Pre každý roh má systém uložené 3 súradnice a to vzdialenosť od senzora *Kinect* v milimetroch, súradnice na ose x v pixeloch a hranicu y, po ktorú je možné dotyk detekovať, ktorá je reprezentovaná poradím riadka v prijatej snímke. Systém má teda súradnice plochy v tvare lichobežníka, ktorá reprezentuje plochu premietaného obrazu na stene. Na základe týchto súradníc systém vypočíta súradnice jednotlivých hraníc, ktoré budú snímať dotyky. Každá z hraníc sníma iba určitú časť plochy, ktorej veľkosť je možné meniť. Počet hraníc sa odvíja od veľkosti snímanej plochy pre každú z nich. Čím je plocha, ktorú hranica sníma väčšia, tým je celkový počet hraníc menší.

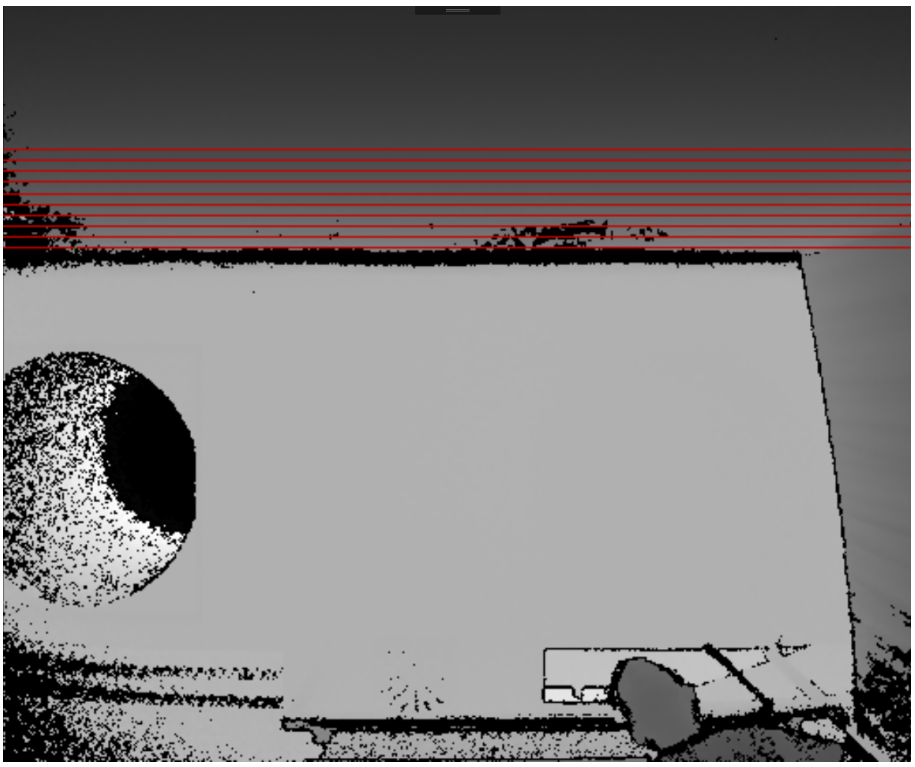
```
int indexDifference = topRowIndex - bottomRowIndex;
int indexShift = indexDifference / numOfTouchZones;
double indexCalibration = Convert.ToDouble(indexDifference %
    numOfTouchZones) / Convert.ToDouble(indexShift);
...

for (int i= 0; i<= numOfTouchZones; i++)
{
   int newIndex = bottomRowIndex + i * indexShift +
       Convert.ToInt16(indexCalibration * (i + 1)) + 1;
   int newDepth = Convert.ToInt16(bottomDepth + (i - 1) * (TOUCHZONE_SIZE +
       calibrationDepth));
```

```
        int xLeft = (lowerLeft.xPixels + Convert.ToInt16(i * xLeftShift));
        int xRight = (lowerRight.xPixels - Convert.ToInt16(i * xRightShift));
        this.touchZones.Add(new TouchZone(newIndex, newDepth, xLeft, xRight));
    }
```

Ukážka kódu naznačuje, ako výpočet jednotlivých hraníc prebieha. V prvej časti je ukážka výpočtu jednotlivých premenných, ostatné premenné sú vypočítané obdobne. Premenná *bottomRowIndex* uchováva hodnotu poradia riadka v prijatej snímke, v ktorej bol detekovaný dotyk v spodnej časti premietaného obrazu, *bottomDepth* obsahuje hĺbku v tomto riadku (teda vzdialenosť od senzora *Kinect*). *TOUCHZONE_SIZE* je voliteľná konštanta, ktorá udáva výšku hranice a teda jej vertikálny rozsah. Na základe jej veľkosti a výšky snímaného obrazu je vypočítaný počet potrebných hraníc, ktorý udáva premenná *numOfTouchZones*. Premenné *xLeft* a *xRight* si uchovávajú hodnotu vzdialenosti hranice od ľavého okraja snímky, ktorá je udávaná v pixeloch. Po výpočte všetkých hodnôt je vytvorená nová hranica, ktorá je uložená do zoznamu hraníc. Jedná sa iba o ukážku výpočtu, celý výpočet je možné vidieť v prílohe tejto práce.

Pripravený systém

Po inicializácii, ktorá bola opísaná v predchádzajúcej podsekcii je systém pripravený na používanie. Inicializovaný systém môže vyzerať tak ako ukazuje nasledujúci obrázok 13.



Obrázok 13: Inicializovaný systém.

Ako bolo vysvetlené, počet snímaných hraníc nie je pevne daný. Ich počet závisí na výške premietaného obrazu, vertikálnej vzdialenosti tohto obrazu od senzora a stanovenej výšky hranice. Pixely, ktoré systém kontroluje na detekciu dotykov sú vyznačené červenou farbou. Žiadne iné pixely systém pre funkčnosť nepotrebuje. Celkový počet pixelov, ktorý systém spracováva je násobok počtu hraníc a čísla 512, čo je šírka prijatej snímky v pixeloch. Výpočtová náročnosť je teda výrazne znížená.

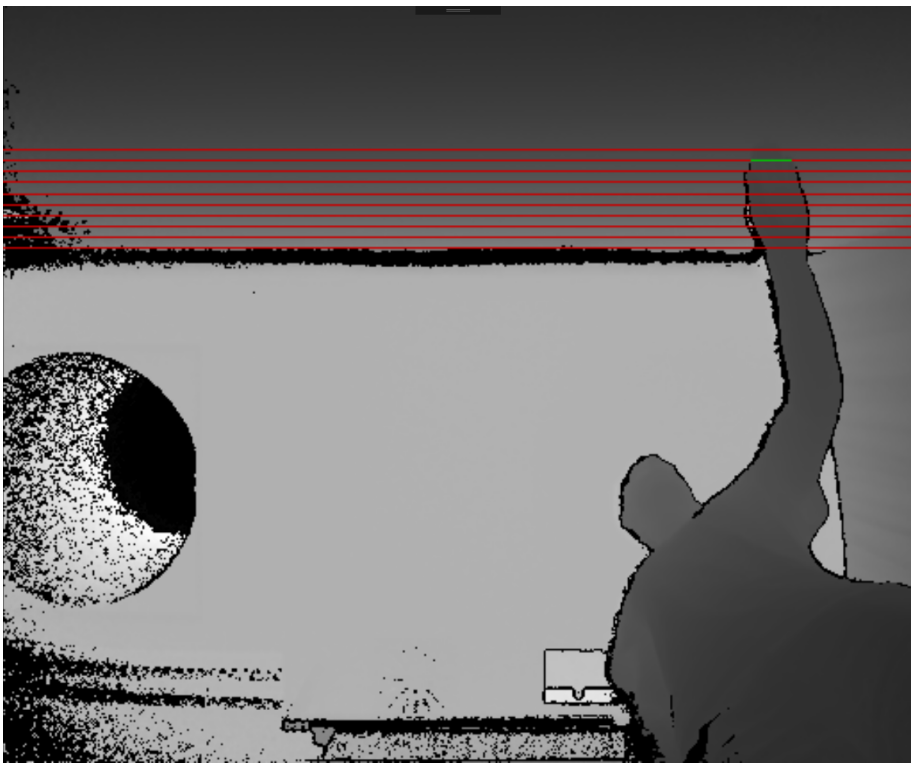
```
foreach (TouchZone touchZone in this.touchZones)
{
    this.isThereTouchInRow(this.touchZones.IndexOf(touchZone));
}
```

Pri príchode novej hĺbkovej snímky zo senzora sa prekontroluje každá z hraníc, ktorá bola vytvorená pri inicializácii systému. Parametrom metódy *isThereTouchInRow* je index kontrolovanej *touchZone* v uloženom poli. Na prvý pohľad možno zbytočné a lepšie by bolo predávanie celého objektu. Metóda *isThereTouchInRow* však zastrešuje aj inicializáciu systému, kedy je implementácia trochu odlišná a parametrom je priamo poradie kontrolovaného riadka v prijatej snímke.

Detekcia dotyku

V metóde *isThereTouchInRow* už prebieha samotná kontrola systému na dotyk od užívateľa. Postupne sa teda prechádza pixel po pixely v danom riadku. Informácia, v ktorom riadku má kontrola prebehnúť nesie objekt triedy *TouchZone*. Najprv sa skontroluje validita pixela a teda či jeho hodnota nepresahuje rozsah hĺbkového senzora *Kinect*. Ak je pixel validný, skontroluje sa rozdiel hodnoty nového pixela a korešpondujúceho pixela z uloženej hĺbkovej mapy snímanej scény. Hodnota pixelov v hĺbkovom obraze nie je presná, odchylka býva pomerne vysoká. Preto sa následne kontroluje veľkosť spomínaného rozdielu hodnôt. V prípade, že je rozdiel menší, ako stanovené minimum, systém vyhodnotí, že sa nejedná o dostatočnú zmenu, nepovažuje zmenu za dotyk. V opačnom prípade systém začne kontrolovať ďalšie pixely s predpokladom, že prvý pixel vyhodnotený ako dotyk budú nasledovať ďalšie.

Systém obdobným spôsobom kontroluje nasledujúce pixely až kým nenarazí na X pixelov v rade, ktoré nevykazujú známky dotyku. Znamená to, že ak systém označí 4 pixely ako dotyk, nasledovať budú 3 pixely bez dotyku a opäť 4 pixely vyhodnotené ako dotyk, systém bude tieto dve oddelené oblasti považovať za jeden dotyk (za predpokladu, že 3 je menšie ako X). Po identifikácii viac ako X pixelov bez dotyku systém pokračuje v prehľadávaní riadka. Ak však opäť narazí na pixel, ktorý vykazuje známky dotyku, je tento dotyk považovaný za druhý (nový) dotyk na tejto hranici. V situácii, že je počet pixelov v jednom dotyku menší ako Y, je tento dotyk považovaný za nevalidný a systém ho ignoruje. Toto opatrenie je z dôvodu nepresnosti senzora, kedy vznikali situácie, že sa objavili 1 či 2 pixely na niektorej z hraníc, ktoré boli vyhodnotené ako dotyky (bez interakcie užívateľa).



Obrázok 14: Dotyk v spodnej časti premietaného obrazu.

Nielen zmena hĺbky je rozhodujúcim faktorom pre jednotlivé hranice. Každá hranica totižto kontroluje iba určitý rozsah hĺbky. Preto ak je prijatý pixel mimo tohto rozsahu, systém ho označí ako pixel bez identifikácie dotyku. Názorný príklad môžeme vidieť na obrázku 14, na ktorom je zachytený dotyk v spodnej časti obrazovky. Vidíme, že dotyk je identifikovaný presne na tej hranici, ktorá sa stará o daný rozsah hĺbky. Vďaka takémuto rozdeleniu hĺbky pre jednotlivé hranice nedochádza k identifikácii dotyku na vyššie položených hraniciach, napriek tomu, že ruka prešla cez túto hranicu skôr, ako cez tú, ku ktorej tento dotyk patrí.

Ku každému takto identifikovanému dotyku si systém uloží štruktúru typu *FrameTouch*. Táto štruktúra si drží informácie o indexoch na ose x, údaj o priemernej hĺbke dotyku (priemer hĺbky pixelov, ktoré boli identifikované ako dotyk) a poradie riadka v prijatej snímke a teda pozíciu na ose y.

Kontrola umiestnenia dotyku na ose x zatiaľ v tomto kroku neprebieha. Tá prebieha až v ďalšej metóde *updateTouch*, ktorá je volaná po dokončení kontroly všetkých pixelov na danej hranici.

Spomínaný postup je vykonávaný pre každý prijatý snímok z hĺbkového senzora, v tomto prípade teda 30× za sekundu. Po každej kontrole hranice je v prípade detekcie dotyku kontrolovaný aj počet snímiek, v ktorom bol dotyk detekovaný. V prípade, že je tento počet párny, zavolá sa metóda *updateTouch*. Dôvodom, prečo sa volá *updateTouch* každú druhú snímku je, že s väčším počtom údajov o doty-

ku (v tomto prípade z dvoch snímiek) sa zvyšuje presnosť určenia lokácie dotyku a prebieha ďalšia eliminácia nechcených dotykov. Nejedná sa o citlivé spomalenie detekcie dotykov, či neskorá aktualizácia dotyku. Pri 30 snímkach za sekundu je doba prijatia dvoch snímiek približne 0,07 sekundy, čo je pre túto interaktívnu plochu za cenu zvýšenia presnosti prijateľná hodnota.

Výpočet súradníc dotyku

Metóda *updateTouch*, ktorá bola spomenutá už viackrát, má na starosti aktualizáciu dotyku, čomu napovedá samotný názov funkcie. Parametrami sú informácie o dotyku v posledných dvoch snímkach a hranica, v ktorej bol dotyk detekovaný. Na základe informácii o dotyku, ktoré tvoria dve štruktúry (každá pre jednu snímku) typu *FrameTouch* sú vypočítané súradnice dotyku.

```
calibrated_yDepth = (depthSum / numOfTouchPixels);
calibrated_xPixels = (indexesSum / numOfTouchPixels);

double screenX = ((double)(calibrated_xPixels - touchZone.xLeft) /
                  touchZone.pixelWidth()) * this.screenResolutionX;
double screenY = this.screenResolutionY -
                  (((double)(calibrated_yDepth - this.bottomRowDepth) /
                  this.heightDifference()) * this.screenResolutionY);
```

Pre súradnicu x sa využíva totožná súradnica a to x v prijatej snímke. Túto súradnicu získame výpočtom priemernej súradnice x v detekovanom dotyku, čoho výsledkom je *calibrated_xPixels*. V tomto momente sa využije informácia o tom, ktorá hranica tento dotyk detekovala. Dôvodom je, že systém potrebuje vedieť, aká šírka obrazu korešponduje s miestom, kde bol dotyk detekovaný. V návrhu riešenia je vysvetlené, že na hĺbkovom obraze sa premietaný obraz javí ako lichobežník. Aby systém vedel určiť výslednú súradnicu x na premietanom obraze, musí pomerovo previesť lokáciu dotyku na tejto hranici k aktuálnej šírke premietaného obrazu v pixeloch (*screenX*).

Súradnica y dotyku na premietanej ploche je vypočítaná pomocou údaja o vzdialenosti dotyku ruky od senzora. Táto hodnota je vypočítaná ako priemerná hĺbka pixelov, na ktorých bol dotyk detekovaný (*calibrated_yDepth*). Metóda *heightDifference* vracia výšku premietaného obrazu v mm. Údaj má systém v milimetroch, ale podľa inicializačných hodnôt štyroch rohov je pomerne jednoduché získať túto hodnotu v pixeloch pre umiestnenie na premietanom obraze.

Takto systém získa súradnice x a y dotyku. V tomto momente je systém pripravený predať informáciu o dotyku triede, ktorá sa stará o interpretáciu týchto dotykov. Najprv však získa informácie, či je na ploche nejaký aktívny dotyk. Ak nie, jednoducho zavolá triednu metódu *beginTouch*, ktorá patrí triede *Events*, aby začala nový dotyk. V opačnom prípade, teda ak je nejaký dotyk aktívny, požiada triedu *Events* o informácie o tomto aktívnom dotyku. Porovná vypočítané súradnice detekovaného dotyku a dotyku, ktorý je už aktívny a vyhodnotí, akú akciu vyvo-

lať. Nastať môžu iba dve situácie. Prvou z nich je, že súradnice dotykov sú príliš odlišné. Systém teda vyvolá akciu začiatku nového dotyku. Druhá situácia je, že rozdiel súradníc je malý a detekovaný dotyk je považovaný za ten istý, ktorý je už aktívny. V takom prípade sa systém na základe veľkosti rozdielu rozhodne, či dotyk aktualizuje, alebo ho nebude vôbec meniť. Dôvod, prečo by systém dotyk vôbec neaktualizoval je viackrát spomínaná nepresnosť senzora, ktorá bez tejto kontroly spôsobuje rýchle kmitanie dotykov na obrazovke (v jednotkách pixelov).

Okrem zaslania informácii pre triedu *Events* si systém uloží informácie o dotyku aj pre svoje potreby. Hlavným dôvodom je, aby systém vedel, kedy je potrebné dotyk ukončiť. Aktívne dotyky si ukladá v dátovom type *List<TouchDetectionTouch>*. Každý objekt typu *TouchDetectionTouch* si uchováva informáciu o počte snímiek, v ktorých sa dotyk vyskytuje, o počte snímiek, v ktorých sa dotyk vyskytuje bez zmeny a o počte snímiek, v ktorých sa už dotyk nenachádza.

Kontrola týchto hodnôt prebieha vždy po prijatí novej snímky a kontrole systému pre dotyky na jednotlivých hraniciach. Ak je počet snímiek, v ktorom sa dotyk vyskytuje bez zmeny väčší ako 30 (čo je približne 1 sekunda), systém zašle informáciu pre triedu *Events* o tom, že je potrebné vykonať *longpress*. V prípade, že je počet snímiek, v ktorých sa už dotyk nenachádza väčší ako 8 (približne 0,25 sekundy), dotyk je ukončený.

Dotyky sa ukladajú globálne a nie ku každej hranici zvlášť, najmä kvôli jednoduchšej práci s nimi. Konkrétne ide o prípad, kedy prebieha vertikálny posun po pracovnej ploche. Vtedy prechádza interakcia od užívateľa cez viaceré hranice a predávanie dotyku medzi nimi by bolo zložité a zbytočné.

Akcia dotyku

Ako bolo spomínané, po určení súradníc dotyku je tento dotyk predaný statickej triede *Events*. Táto trieda má na starosti vykonať príslušnú akciu pre každý dotyk. Tejto triede sa nepredá iba samotný dotyk, trieda dostane aj inštrukciu, ktorú má vykonať a teda či dotyk začať, aktualizovať alebo ukončiť. Je to preto, že o tom rozhoduje samotný systém, nie trieda *Events*, ktorá však implementuje konkrétne akcie.

Je známe, že *Windows 10*, pre ktorý je tento systém programovaný, je pripravený na interakciu dotykmi, čomu nasvedčuje jeho nasadenie v mobilných telefónoch a tabletoch. Preto bola snaha pre jedného užívateľa využiť túto možnosť a poskytnúť mu interakciu naprieč celým systémom, nielen v konkrétnej aplikácii. Po viacerých vlastných pokusoch a testoch cudzích frameworkov bol vybraný *TCD.System.TouchInjection*, ktorý je dostupný aj cez správcu balíkov *NuGet*.

Vybraný framework umožňuje jednoduché vytvorenie vlastného dotyku a následne tento dotyk vyslať priamo do systému *Windows*. Ten nedokáže rozlíšiť takto vytvorený dotyk a skutočný dotyk na dotykovej obrazovke. Dotyk obsahuje veľké množstvo atributov, hlavnými z nich sú však súradnice x a y.

```
TouchInjector.InjectTouchInput(1, new[] { this.pointer });
```

Zaslanie informácie o dotyku do systému je pomerne jednoduché. Pre tento účel slúži statická metóda *InjectTouchInput*. Parametrami sú počet vkladaných dotykov (*1*) a pole objektov triedy *PointerTouchInfo* (*new[] this.pointer*), ktoré nesú informácie o dotyku. Systém *Windows* nemá podporu pre viac ako jeden dotyk súčasne. Parameter pre počet vkladaných dotykov slúži pre gestá (napr. *pinch to zoom*), kde sa jedná vlastne o dva dotyky zároveň, avšak vždy len pre jednú akciu.

Či už sa jedná o začatie nového dotyku, aktualizáciu alebo ukončenie aktívneho dotyku, volá sa vždy métoda z predchádzajúcej ukážky. Rozdielne sú však atributy, ktoré nesie objekt, konkrétne *this.pointer* typu *PointerTouchInfo*. Tento objekt obsahuje tzv. indikátor, ktorý systému definuje požadovaný stav. V nasledujúcej ukážke kódu vidíme tri rôzne varianty týchto stavov.

```
//indikátor pri vytvorení dotyku
this.pointer.PointerInfo.PointerFlags = PointerFlags.DOWN |
    PointerFlags.INRANGE | PointerFlags.INCONTACT;

//indikátor pri aktualizácii
this.pointer.PointerInfo.PointerFlags = PointerFlags.UPDATE |
    PointerFlags.INRANGE | PointerFlags.INCONTACT;

//indikátor pri ukoncení dotyku
this.pointer.PointerInfo.PointerFlags = PointerFlags.UP;
```

Trieda *Events* si drží zoznam aktívnych dotykov globálne v dátovom type *List<PointerTouch>* resp. *List<SimpleTouch>*. Pri aktualizácii dotyku sa totižto použije objekt, ktorý bol vytvorený pri začatí dotyku. Objektu sa zmenia atributy definujúce indikátor a súradnice. Zmeny týchto dvoch atribútov postačujú na aktualizáciu dotyku. V prípade ukončenia dotyku stačí iba zmena indikátora.

V odstavci o zaslaní informácie o dotyku do systému je zmienka o tom, že systém *Windows* nemá podporu pre viac simultánnych dotykov. Tento fakt zabraňuje využívať celý systém *Windows* viacerým užívateľom zároveň. Preto je nutné pre akúkoľvek kolaboratívnu prácu implementovať vlastnú aplikáciu, za predpokladu, že je nutné, aby užívatelia pracovali s pracovnou plochou zároveň. Je zrejmé, že existujú prípady, pre ktoré to nutné nie je a jedná sa síce o interakciu viacerých užívateľov, ale nie zároveň. Príkladom môže byť hra pexeso, kde sú zúčastnení viacerí hráči, avšak s plochou pracuje vždy len jeden z nich.

Pre potrebu interakcie viacerých užívateľov súčasne je teda nutné dotyky riešiť vlastnoručne. Aby to bolo možné, trieda *Events* zasiela informácie buď priamo do operačného systému, alebo uskutočnené udalosti posiela zvolenému príjemcovi. Za týmto účelom zasiela trieda udalosti uvedené v nasledujúcom kóde (jedná sa iba o výpis metód).

```
touchHasBegun (uint touchId, Point location)

longpressOccured(uint touchId, Point location);

touchMoved(uint touchId, Point location)
```

```
touchHasEnded(uint touchId)
```

Tieto štyri udalosti by mali byť dostačujúce pre akúkoľvek jednoduchú interakciu s pracovnou plochou. Systém na detekciu dotykov je teda samostatný a nie je priamo viazaný na testovaciu aplikáciu. Ktokoľvek by si chcel implementovať vlastnú aplikáciu, môže tak učiniť bez zásahu do systému na detekciu dotykov. Musel by však implementovať vyššie uvedené udalosti, aby interakcia užívateľa s pracovnou plochou fungovala.

5.4 Implementácia testovacej aplikácie

Predchádzajúci text obsahoval informáciu, prečo nie je možné využívať operačný systém pre interakciu s viacerými užívateľmi naraz. Interakciu viacerých užívateľov zároveň je teda nutné spracovať vlastnoručne. Okrem implementácie funkcií vlastného programu je nevyhnutné implementovať reakcie na akcie, ktoré boli predstavené v predchádzajúcej podkapitole. Skôr, než bude opísaná implementácia týchto udalostí, bude podrobnejšie opísaná testovacia aplikácia.

Hlavným účelom tejto testovacej aplikácie je overiť funkčnosť implementovaného systému na detekciu dotykov. Pri vytváraní návrhu sa kladol dôraz najmä na to, aby poskytovala vopred požadovanú funkcionalitu. Konkrétne sa jedná o rozdelenie pracovnej plochy na personalizované oblasti, možnosť predať obsah medzi užívateľmi (ich pracovnými plochami) a možnosť prispôsobiť vzhľad zobrazovaných objektov (napr. zmena pozície).

Návrh aplikácie bol opísaný spolu s návrhom systému na detekciu dotykov v podsekcii *Finálny návrh*. V tejto sekcii bude predstavená už implementovaná aplikácia.

Obrázok 15: Hlavná scéna testovacej aplikácie.

Na obrázku 15 je znázornená hlavná scéna testovacej aplikácie, ktorá sa objaví ihneď po zapnutí aplikácie. V tomto príklade je plocha rozdelená na dve pracovné oblasti, ktoré sú farebne odlíšené. Tretia plocha v spodnej časti obrazovky, ktorá zaberá celú šírku obrazovky, je spoločná vždy pre všetkých užívateľov.

V personalizovaných oblastiach vidí každý užívateľ tweety zo svojho vlastného *Twitter* účtu. Systém si ukladá informácie o prihlásených užívateľoch, preto nie je potrebné prihlasovať sa pri každom štarte aplikácie. Ak sa aplikácia spustí pre viac užívateľov, než má aplikácia uložených, na pracovnej ploche tohto užívateľa sa nezobrazí žiaden tweet, ale tlačidlo pre prihlásenie užívateľa. Keďže sa jedná iba o testovaciu aplikáciu, nie aplikáciu určenú pre verejnosť, prihlasovanie jednotlivých užívateľov prebieha samostatným procesom, ktorý nie je pre užívateľa dostupný, viditeľné je iba spomínané tlačidlo prihlásiť, ktoré má za úlohu uľahčiť prácu správcovi aplikácie.

Po načítaní tweetov pre každého užívateľa je aplikácia pripravená k používaniu. Každý z užívateľov si môže prehliadať tweety na svojej pracovnej ploche. Každá pracovná plocha môže obsahovať viac tweetov, ako sa vmestí na obrazovku, v aplikácii funguje *scroll*, nezávisle pre každého užívateľa. *Scroll* funguje tak ako na mobilných zariadeniach, teda posunom dotyku po pracovnej ploche.

Pri jednoduchom dotyku na akýkoľvek tweet sa vyvolá akcia zobraziť detail,

tweet sa zväčší a zobrazí v strede pracovnej plochy užívateľa. Do zobrazeného detailu sú pridané informácie, ktoré predtým zobrazené neboli ako prezývka užívateľa, ktorý daný tweet pridal, či dátum, kedy tweet vznikol. Zatvorenie detailu sa vykonáva rovankým spôsobom ako jeho otvorenie a to jednoduchým dotykom na obrazovku.

Obrázok 16: Zobrazený detail vybraného tweetu.

Aplikácia ďalej umožňuje presun jednotlivých tweetov medzi užívateľmi. Aby užívateľ zahájil presun tweetu, musí vykonať *longpress* na tweete, ktorý chce presunúť. *Longpress* aktivuje vybraný tweet na presun. Tweet je zvýraznený a umiestnený v popredí aplikácie. Túto akciu znázorňuje nasledujúci obrázok 17.

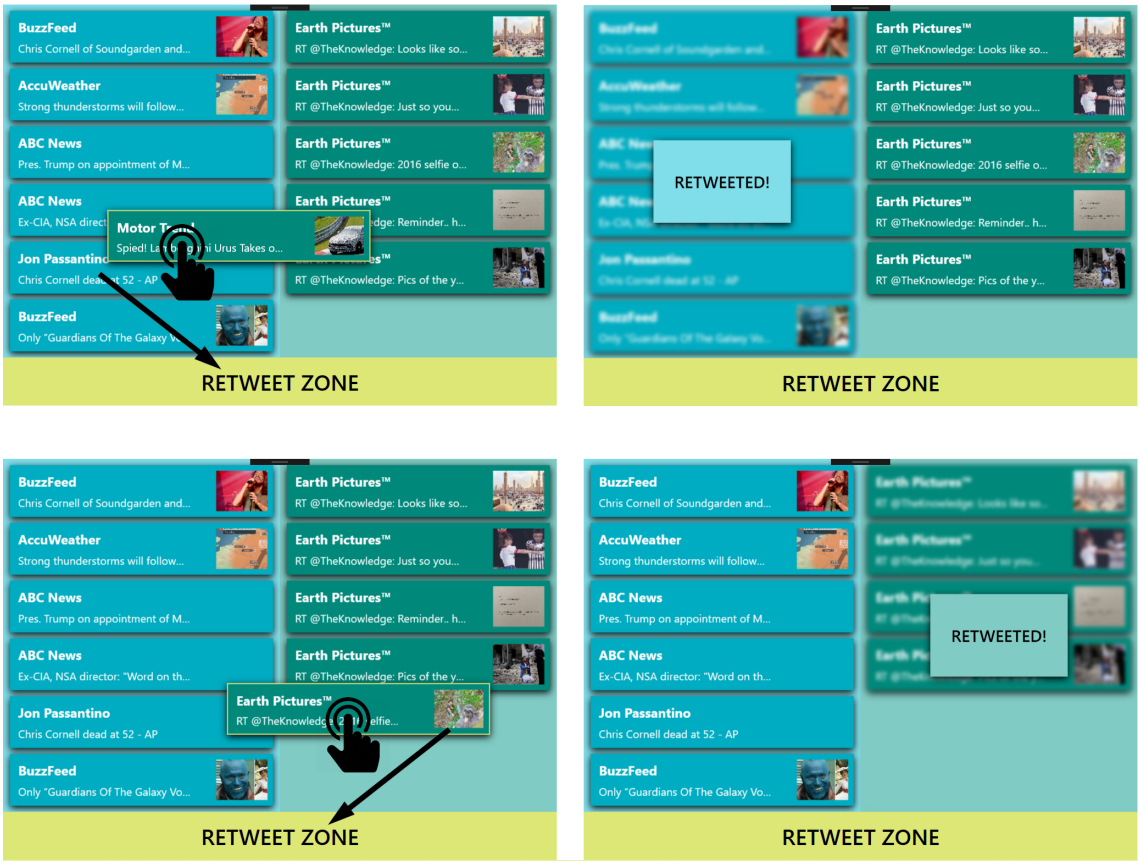
Obrázok 17: Užívateľ presúva tweet medzi pracovnými plochami.

Lokácia, v ktorej bol dotyk ukončený určuje, ku ktorej pracovnej ploche sa tweet priradí. Tweet sa následne zaradí medzi už existujúce tweety, a to na prvé miesto. Ostatné tweety sa posunú o jednu pozíciu nižšie. Presun tweetu zo susednej pracovnej plochy na svoju umožňuje retweetnuť tento tweet zo svojho vlastného účtu. Retweet zóna totižto prepošle prijatý tweet za toho užívateľa, z ktorého pracovnej plochy bol tento tweet vzatý, nie za toho užívateľa, z ktorého domovskej stránky bol tento tweet stiahnutý.

Pre príklad poslúži obrázok 18. Na prvej pracovnej ploche (vľavo) je možné vidieť tweet, ktorý bol presunutý z pracovnej plochy druhého užívateľa. Ak bude nasledujúcim krokom presun tohto tweetu do retweet zóny, autorom retweetu bude užívateľ prvej pracovnej plochy napriek tomu, že tento tweet pochádza od užívateľa druhej pracovnej plochy. Po úspešnom retweete sa zobrazí potvrdzujúca hláška o retweetnutí na pracovnej ploche toho užívateľa, z ktorého účtu bol daný retweet vykonaný (v tomto prípade na pracovnej ploche prvého užívateľa).

Obrázok 18: Presunutý tweet na susednú pracovnú plochu.

Akcie, ktoré sú opísané v predchádzajúcom odstavci, sú znázornené na nasledujúcom obrázku 19, ktorý nadväzuje na predchádzajúci obrázok 18.



Obrázok 19: Ukážka retweetu jednotlivých užívateľov.

Aplikácia je na pohľad pomerne jednoduchá. Je však potrebné si uvedomiť, že musí byť implementovaná celá interakcia užívateľa s touto aplikáciou. Do aplikácie sa dostanú iba spomínané štyri udalosti, ktoré posiela trieda *Events*. To, ako si ich aplikácia spracuje, je nutné ošetriť samostatne.

Aby mohlo byť opísané, ako aplikácia tieto udalosti rieši, bude najskôr priblížená jej štruktúra. Aplikácia sa spustí vždy v samostatnom okne. Pred spustením je zahájená detekcia počtu užívateľov. Každý užívateľ sa prezentuje priložením ruky na premietanú plochu. Podľa zisteného počtu užívateľov sa vytvorí príslušný počet užívateľských plôch (objektov triedy *UserDesktop*). Hlavné okno taktiež obsahuje retweet zónu v spodnej časti obrazovky (objekt triedy *RetweetZone*). Každá z užívateľských plôch následne obsahuje bunku (objekty triedy *TweetCell*), ktorá obsahuje ďalšie elementárne prvky ako nadpis, text a obrázok, ktoré však nie je nutné popisovať.

Prvú udalosť, ktorú môže aplikácia dostať je *touchHasBegun*. Všetky udalosti, či už to bude longpress, alebo pohyb dotyku po obrazovke musia najprv začať. V tomto momente si aplikácia uloží dotyk, ktorý prišiel ako parameter tejto metódy do svojho zoznamu aktívnych dotykov.

Vzápätí môže prísť akcia *longpressOccured*. V takomto prípade má aplikácia za

úlohu aktivovať príslušný tweet a pripraviť ho na presun po ploche. Parametrom aj tejto metódy je *id* dotyku a jeho lokácia. Podľa lokácie sa aplikácia pokúsi vyhľadať *TweetCell*, ktorý sa na tejto lokácii nachádza.

```
UserDesktop desktop = this.findCorrespondingDesktop(location);
if (desktop != null)
{
   TweetCell tweetCell = desktop.tweetCellAtLocation(location);
   if (tweetCell != null)
   { ... }
}
```

Za predpokladu, že sa na danej lokácii nejaký objekt triedy *TweetCell* nachádza, je tento objekt presunutý do popredia a uložený do zoznamu aktívnych. Znamená to, že už nepatrí pod *UserDesktop* ale priamo pod celé okno aplikácie. Presun tohto objektu do najvrchnejšej vrstvy umožňuje presúvať tento objekt aj medzi plochami ostatných užívateľov. Ak by tento objekt stále patril pod niektorý z objektov triedy *UserDesktop*, nebolo by možné presunúť ho na žiadny iný, pretože jeho súradnice by nadobúdali nevalidné hodnoty, keďže ostatné užívateľské plochy sa v zobrazovacej hierarchii nachádzajú o úroveň vyššie.

Po akcii *touchHasBegun* môže nastať akcia *touchMoved*. Rovnako ako predchádzajúce dve metódy, aj táto má dva parametre a to *id* dotyku a jeho lokáciu. Aplikácia dostane informáciu, že niektorý z dotykov zmenil lokáciu a podľa *id* vyhľadá tento dotyk v svojom zozname aktívnych dotykov. Zároveň skontroluje, či k danému dotyku nemá taktiež uložený aktívny objekt *TweetCell*. Podľa toho, či k tomuto dotyku aktívny objekt existuje, je buď vykonaný presun tohto objektu, alebo uskutočnený *scroll* na príslušnej užívateľskej ploche.

Súradnice novej pozície aktívneho objektu *TweetCell* sa vypočítajú ako rozdiel pozície uloženého dotyku a pozície dotyku, ktorý prišiel v metóde *touchMoved*. Rovnakým spôsobom sa počíta aj hodnota, o veľkosť ktorej sa má vykonať prípadný *scroll* obrazovky, v tomto prípade sa však počíta iba s osou y, keďže aplikácia neumožňuje horizontálny posun plochy.

Posledná akcia, ktorú môže aplikácia dostať je *touchHasEnded*. Aplikácia si opäť nájde dotyk v zozname uložených dotykov a k nemu prislúchajúci aktívny prvok. V prípade, že dotyk naozaj aktívny prvok má, aplikácia zistí, čo sa na mieste ukončenia dotyku nachádza, konkrétne to môže byť niektorá z užívateľských plôch, alebo retweet zóna. Môžu nastať teda dve situácie a to umiestnenie aktívneho objektu *TweetCell* do prislúchajúcej užívateľskej plochy, alebo retweet za užívateľa, z ktorého pracovnej plochy tento tweet pochádza. Atribút, ktorý tento údaj obsahuje si drží samotný objekt *TweetCell* a tak je pomerne jednoduché tento retweet vykonať.

V popise aplikácie sa uvádza zobrazenie detailu vybraného tweetu, ktorý sa vyvolá jednoduchým dotykom na obrazovku. Takáto udalosť však do aplikácie nepríde. Tento stav je nutné detekovať priamo v aplikácii. Jednoduchý dotyk (nahrádza klik na myši) na určitých súradniciach začne a na rovnakých aj skončí. Zároveň ne-

smie byť príliš dlhý, aby sa neaktivoval *longpress*. Preto tento účel si aplikácia drží zoznam dotykov, ktoré zmenili svoju polohu, či aktivovali longpress. Pri ukončení každého dotyku je tento zoznam skontrolovaný a v prípade, že neobsahuje dotyk, ktorý práve skončil, vyvolá sa akcia kliku. Rovnako ako v prípade akcie *longpress* sa vyhľadá príslušný tweet, tentokrát sa však neaktivuje na presun, ale zobrazí sa jeho detail.

Aplikácia pre prácu so sociálnou sieťou *Twitter* používa framework *Tweetinvi*. Framework uľahčuje prácu s *Twitter* API a ponúka možnosť pracovať s ním takmer okamžite, bez nutnosti zdĺhavého študovania dokumentácie oficiálneho API.

6 Diskusia

Vytvorený systém na detekciu dotykov a k nemu príslušná aplikácia poskytuje ukážku, že interaktívnu stenu je možné vytvoriť z akejkoľvek bežnej steny, projektora a hĺbkového senzora. V tomto prípade bol použitý senzor *Microsoft Kinect*, avšak samotný princíp fungovania systému naň nie je viazaný. Preto by po menších úpravách (komunikácia so senzorom) bolo možné použiť akýkoľvek iný senzor, ktorý dokáže poskytnúť hĺbkový obraz snímanej scény. Tento fakt vnímam ako kladný, keďže v prípade, že niekto disponuje hĺbkovým senzorom, nemusí kupovať práve *Microsoft Kinect*, ale systém bude fungovať aj s jeho senzorom.

Za jednu z výhod, ktorými navrhnutý systém disponuje považujem obstarávaciu cenu. Cieľovou skupinou týchto interaktívnych plôch sú prevažne školy a rôzne pracoviská. Väčšina subjektov, ktoré si takúto plochu plánujú zabezpečiť už s veľkou pravdepodobnosťou počítač a projektor vlastnia. Postačí preto, aby zakúpili hĺbkový senzor a dokážu vytvoriť interaktívnu stenu s minimálnymi nákladmi v porovnaní s kúpou celej interaktívnej tabule či iného riešenia.

Systém, ktorý bol vytvorený však zatiaľ nie je pripravený na komerčné využitie. Niektoré nastavenia je potrebné meniť priamo v zdrojovom kóde. Cieľom práce však nebolo vytvoriť aplikáciu, ktorá bude na komerčné využitie pripravená. Napriek tomu si myslím, že je systém veľmi dobre pripravený na to, aby mohol byť využiteľný aj v tejto sfére. Aby bolo možné tento systém takto verejne poskytovať, vyžadoval by si niekoľko úprav, avšak celá logika systému by ostala nezmenená.

Je veľká škoda, že operačný systém *Windows 10* nepodporuje interakciu viacerých užívateľov zároveň. Z tohto pohľadu je tvorba akejkoľvek, aj na pohľad jednoduchej aplikácie pomerne náročná. Bez vlastnej implementácie nefungujú žiadne interaktívne prvky, ktoré môže aplikácia obsahovať (tlačidlá, *scroll* atď.). Tento stav by sa dal vyriešiť akousi samostatnou nadstavbou, ktorá by po identifikácii dotyku vyhľadala v aplikácii na danej lokácii prvok a v prípade, že sa jedná o interaktívny prvok, vyslala by udalosť, ktorá pre daný prvok patrí. Takto by sa ušetrila práca pre budúce aplikácie, avšak iba na systémové prvky. V prípade potreby špeciálneho prvku, alebo vlastného chovania by bola vyžadovaná vlastná implementácia.

Navrhnuté riešenie má isté technické obmedzenia, ktoré sa však dajú zredukovať. Jedným z obmedzení je, že senzor musí byť umiestnený rovnobežne so stenou, keďže jedna zo súradníc je valstne poradie riadka v prijatej snímke hĺbkového obrazu. Ak by senzor nebol umiestnený so stenou rovnobežne, systém fungovať síce bude, ale bude dochádzať k príliš skorým dotykom a príliš neskorej odozve. Obmedzenie by sa dalo vyriešiť rovnako ako je riešené skreslené videnie senzora (kvôli FOV) a to vytvorením viacerých hraníc. Vznikla by tak akási mriežka na detekciu dotykov. Toto obmedzenie systému však pri správnom umiestnení senzora nemá na jeho funkčnosť vplyv.

Ako bolo spomenuté pri návrhu systému, hrana medzi stenou a stropom spôsobuje senzoru *Kinect* problémy a nedokáže v týchto miestach namerať hĺbku. Preto je nutné pri umiestňovaní senzora na tento fakt myslieť a toto umiestnenie prispôso-

biť tak, aby tento šum nespôsoboval nesprávnu detekciu dotykov. Túto nepresnosť v meraní považujem za najväčšie technické obmedzenie pre navrhnutý systém.

Umiestnenie hĺbkového senzora je pod stenou. To znamená, že určitým spôsobom môže prekážať pri interakcii s premietaným obrazom. Hlavným dôvodom, prečo bolo vybrané toto umiestnenie je jednoduchá a rýchla príprava. Systém bol vyvíjaný a testovaný v školskom laboratóriu virtuálnych technológií a takéto umiestnenie senzora nevyžadovalo žiadnu špeciálnu prípravu. V praxi by však bolo lepšie, ak by sa senzor nachádzal nad premietaným obrazom a neprekážal by pri interakcii. Takáto zmena umiestnenia by neznamenala potrebu nového návrhu. Jednoducho povedané, stačilo by súradnice dotyku počítať opačne.

Považujem za zaujímavú myšlienku doplniť hĺbkový obraz o iný zdroj, či už RGB, alebo infračervenú kameru. Detekcia dotyku by tak mohla byť uskutočnená naprieč viacerými zdrojmi obrazu a tým by sa zvýšila jej presnosť. Rovnaký dopad by malo aj použitie hĺbkovej kamery s vyšším rozlíšením, avšak súčasný trh takúto kameru s rovnakou technológiou detekcie hĺbky neponúka. Preto by som rád vyskúšal tento systém aj so senzorom *ZED*. Jedná sa síce o odlišnú technológiu, ale výsledky by mohli byť zaujímavé.

Čo sa týka testovacej aplikácie, myslím si, že splnila svoj účel. Aplikácia je síce jednoduchá, ale poskytuje potrebný priestor na overenie funkčnosti implementovaného systému. Implementáciu akejkoľvek aplikácie pre viacerých užívateľov (pre tento systém) považujem za veľmi náročnú. Žiadna interakcia užívateľa totižto nebude fungovať bez toho, aby bola v tejto aplikácii implementovaná.

7 Záver

Cieľom tejto práce bolo navrhnúť koncept inteligentnej pracovnej plochy s použitím senzora *Microsoft Kinect* a implementovať aplikáciu, pomocou ktorej bude funkčnosť tohto konceptu možné overiť. Cieľ práce sa podarilo naplniť, čomu nasvedčujú výsledky práce.

Vykonaná rešerš poskytla dobrý základ pre prácu s hĺbkovým senzorom. Inšpiroval som sa ako komerčnými riešeniami, tak aj výskumnými projektami z oblasti spracovania obrazu hĺbkových senzorov. Aby som do tejto oblasti prispel aj ja, zvolil som si umiestnenie senzora, na ktoré som nenarazil.

V práci je opísaný postup pri návrhu konceptu, jeho finálny návrh a následne aj implementácia. Návrh bol konštruovaný tak, aby umožňoval interakciu viacerých užívateľov zároveň, keďže to bolo jedným z hlavných zameraní tejto práce.

Pre overenie funkčnosti vytvoreného konceptu bola implementovaná aplikácia, ktorá túto možnosť poskytuje. Na prvý pohľad jednoduchá aplikácia s tématikou sociálnej siete *Twitter*. V práci je však opísaná náročná interpretácia každej interakcie užívateľa. Aplikácia umožňuje vykonať všetky z vytýčených úkonov (viacero užívateľov, personalizované oblasti, predanie obsahu medzi užívateľmi, prispôsobenie vzhľadu zobrazovaných objektov) a dokazuje tak funkčnosť navrhnutého systému ako aj samotnej aplikácie.

8 Referencie

AMON, CLEMENS; FUHRMANN, FERDINAND; GRAF, FRANZ.*Evaluation of the spatial resolution accuracy of the face tracking system for kinect for windows v1 and v2*. Proceedings of the 6th Congress of the Alps Adria Acoustics Association [online]. 2014 s. 16-17. [cit. 2017-02-09]. Dostupné z: https://goo.gl/au4YoK.

Andy Wilson at Microsoft Research. Oficiální domovská stránka Microsoft [online]. ©2017 [cit. 2017-02-22]. Dostupné z: https://www.microsoft.com/en-us/research/people/awilson/.

BHALLA, MUDIT RATANA; BHALLA, ANAND VARDHAN.*Comparative study of various touchscreen technologies*. International Journal of Computer Applications [online]. 2010 [cit. 2017-02-08]. Dostupné z: https://goo.gl/G8FFCJ.

BURDEA, GRIGORE. A PHILIPPE. COIFFET.*Virtual reality technology*. 2nd ed. Hoboken, N.J.: J. Wiley-Interscience, c2003. ISBN 0-471-36089-9..

COLLEEN, C.. *What's New in the 2016 R2 Intel RealSense SDK?*. In: Intel® Software [online]. 2016 [cit. 2017-04-02]. Dostupné z: https://software.intel.com/en-us/blogs/2016/05/21/new-r2.

DOWNS, RICK. *Using resistive touch screens for human/machine interface*. In: Proceedings of the International Working Conference on Advanced Visual Interfaces. ACM [online]. 2002, s. 116-123. [cit. 2017-02-08]. Dostupné z: https://goo.gl/lKBsFA.

FRANCESE, RITA; PASSERO, IGNAZIO; TORTORA, GENOVEFFA. *Wiimote and Kinect: gestural user interfaces add a natural third dimension to HCI*. Analog Applications Journal Q [online]. 2005, 3: 5-9. [cit. 2017-02-08]. Dostupné z: https://goo.gl/RaucdA.

HAUBNER, NADIA a kol. *Integrating a depth camera in a tabletop setup for gestural input on and above the surface*. Tech. Rep., Rhein Main University of Applied Sciences [online]. 2013, [cit. 2017-02-18]. Dostupné z: https://goo.gl/UigktX.

HIRSCHMÜLLER, HEIKO; INNOCENT, PETER R.; GARIBALDI, JON. *Real-time correlation-based stereo vision with reduced border errors*. In: International Journal of Computer Vision [online]. 2002, 47.1-3: 229-246. [cit. 2017-04-02]. Dostupné z: https://goo.gl/dUzQS2.

Intel® RealSense™ Camera R200: Embedded Infrared Assisted Stereovision 3D Imaging System with Color Camera. In: Intel® Software [online]. 2016 [cit. 2017-04-02]. Dostupné z:

https://software.intel.com/sites/default/files/managed/d7/a9/realsense-camera-r200-product-datasheet.pdf.

Intel® RealSense™ Developer Kit (R200). Intel [online]. Santa Clara, ©2017 [cit. 2017-04-02]. Dostupné z: https://click.intel.com/intelr-realsensetm-developer-kit-r200-2191.html.

Jing, Pan; Ye-peng, Guan. *Human-computer interaction using pointing gesture based on an adaptive virtual touch screen*. International Journal of Signal Processing, Image Processing and Pattern Recognition, 2013, 6.4: 81-92. [online]. [cit. 2017-03-02]. Dostupné z: http://www.sersc.org/journals/IJSIP/vol6_no4/7.pdf.

Kazmi, Wajahat a kol. *Indoor and outdoor depth imaging of leaves with time-of-flight and stereo vision sensors: Analysis and comparison*. ISPRS journal of photogrammetry and remote sensing, 2014, 88: 128-146. [online]. [cit. 2017-04-02]. Dostupné z: https://upcommons.upc.edu/bitstream/handle/2117/26287/1472-Indoor-and-outdoor-depth-imaging-of-leaves-with-time-of-flight-and-stereo-vision-sensors_-Analysis-and-comparison.pdf.

Kinect hardware. Microsoft – Official Home Page [online]. ©2017 [cit. 2017-02-09]. Dostupné z: https://developer.microsoft.com/en-us/windows/kinect/hardware.

Kinect tools and resources. Microsoft – Official Home Page [online]. ©2017 [cit. 2017-02-08]. Dostupné z: https://developer.microsoft.com/en-us/windows/kinect/tools.

Kramer, Jeff.*Hacking the Kinect*. New York: Apress, c2012. Technology in Action Press book. ISBN 978-1-4302-3867-6..

Kudale, Aniket E. a K.H. Wanjale.*Human Computer Interaction Model based Virtual Whiteboard: A Review*. International Journal of Computer Applications [online]. 2015 [cit. 2017-01-28]. Dostupné z: http://www.ijcaonline.org/research/volume130/number17/kudale-2015-ijca-907203.pdf.

Lachat, E. a kol.*First experiences with Kinect v2 sensor for close range 3D modelling*. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 2015, 40.5: 93..

Libfreenect2: API Reference [online]. 2016 [cit. 2017-03-29]. Dostupné z: https://openkinect.github.io/libfreenect2/.

Mimio Interactive Displays. Innovative Technology in the Classroom | Boxlight Mimio [online]. Lawrenceville, ©2017 [cit. 2017-02-02]. Dostupné z: http://www.mimio.com/en-EM/Products/Interactive-Whiteboards.aspx.

Mimio Price List. Touchboards [online]. Long Branch, ©2016 [cit. 2017-02-02]. Dostupné z: http://www.touchboards.com/mimio/pricelist.asp.

MotionMagix™: Interactive Playground Equipment [online]. Fremont: TouchMagix Media, ©2015 [cit. 2017-02-02]. Dostupné z: http://www.motionmagix.com.

Orbbec: Intelligent computing for everyone everywhere [online]. ©2017 [cit. 2017-04-02]. Dostupné z: https://orbbec3d.com/.

Pagliari, Diana a Livio Pinto. *Calibration of Kinect for Xbox One and Comparison between the Two Generations of Microsoft Sensors* Sensors (14248220) [online]. 2015, 15(11), 27569-27589 [cit. 2017-02-09]. DOI: 10.3390/s151127569. ISSN 14248220..

Promethean ActivWall. Touchboards [online]. Long Branch, ©2016 [cit. 2017-02-02]. Dostupné z: http://www.touchboards.com/promethean/series_activwall.

Promethean Price List. Kernel Software, Inc. - Online technology super eStore [online]. Wausau, 2017 [cit. 2017-02-02]. Dostupné z: http://www.kernelsoftware.com/products/catalog/promethean.html.

Promethean World | Interactive Education Technology for Schools [online]. Alpharetta, ©2017 [cit. 2017-02-02]. Dostupné z: https://www.prometheanworld.com.

Ubi Interactive. Ubi Interactive [online]. Seattle, 2016 [cit. 2017-02-02]. Dostupné z: http://www.ubi-interactive.com.

Vitruvius [online]. New York [cit. 2017-03-29]. Dostupné z: https://vitruviuskinect.com/.

Vitruvius simplifies development of Kinect for Windows apps. Microsoft – Official Home Page [online]. ©2017 [cit. 2017-03-29]. Dostupné z: https://blogs.msdn.microsoft.com/kinectforwindows/2015/11/02/vitruvius-simplifies-development-of-kinect-for-windows-apps/.

Wilson, Andrew D *Using a depth camera as a touch sensor*. In: ACM International Conference on Interactive Tabletops and Surfaces - ITS '10 [online]. New York, New York, USA: ACM Press, 2010, s. 69- [cit. 2017-02-22]. DOI: 10.1145/1936652.1936665. ISBN 9781450303996. Dostupné z: http://portal.acm.org/citation.cfm?doid=1936652.1936665.

ZED - Depth Sensing and Camera Tracking. ZED Stereo Camera [online]. San Francisco, ©2017 [cit. 2017-04-03]. Dostupné z: https://www.stereolabs.com/zed/specs/.

Prílohy

A Priložené CD

Priložené CD obsahuje zdrojový kód systému na detekciu dotykov ako aj zdrojový kód testovacej aplikácie.