Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Bakalářská práce

Využití vstupního zařízení XBOX Kinect

Vypracoval: Jan Předota
Vedoucí práce: Mgr. Radim Remeš

České Budějovice 2015

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta ekonomická
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan PŘEDOTA**
Osobní číslo:	**E12256**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Název tématu:	**Využití vstupního zařízení XBOX Kinect**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování:

Cílem práce je vytvořit software využívající zařízení XBOX Kinect.

Metodický postup:

1. Studium odborné literatury.
2. Publikace výsledků rešerše.
3. Návrh, popis vývoje a implementace aplikace.
4. Zhodnocení, vypracování doporučení a závěrů.

Rozsah grafických prací:

Rozsah pracovní zprávy: **40 - 50 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **WEBB, Jarrett, a Ashley JAMES.** *Beginning Kinect Programming with the Microsoft Kinect SDK.* **Berkely, CA, USA: Apress, 2012. ISBN 978-1430241041.**
2. **CATUHE, David.** *Programming with the Kinect for Windows Software Development Kit.* **Redmont, Washington, USA: Microsoft, 2012. ISBN 978-0735666818.**
3. **JANA, Abhijit.** *Kinect for Windows SDK Programming Guide.* **Birmingham, UK: Packt, 2012. ISBN 978-1849692380.**
4. **MILES, Rob.** *Start Here! Learn the Kinect API.* **Sebastopol, California, USA: O'Reilly (with authorization of Microsoft), 2012. ISBN 978-0735663961.**

Vedoucí bakalářské práce: **Mgr. Radim Remeš**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **7. ledna 2014**

Termín odevzdání bakalářské práce: **15. dubna 2015**

doc. Ing. Ladislav Rolínek, Ph.D.
děkan

prof. RNDr. Pavel Tlustý, CSc.
vedoucí katedry

V Českých Budějovicích dne 26. března 2014

Prohlášení

Prohlašuji, že svoji bakalářskou práci na téma „Využití vstupního zařízení XBOX Kinect“ jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47 zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to - v nezkrácené podobě/v úpravě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou - elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

…… ……

Datum Podpis

Poděkování

Poděkování patří vedoucímu mojí bakalářské práce Mgr. Radimovi Remešovi za velkou ochotu a věcné připomínky, které mi pomohli práci dokončit. Také bych rád poděkoval mojí rodině za podporu, kterou mi při studiu vysoké školy projevují.

Obsah

1 Úvod

S příchodem nových technologií se postupně mění způsob ovládání nejrůznějších zařízení, na který jsme byli doposud zvyklí. Jeden z takových momentů přišel ve chvíli, kdy byla na trh uvedena počítačová myš. Tento okamžik výrazně zjednodušil ovládání a přinesl uživateli způsob, který je pro něj přirozenější. Další revoluce nastala s příchodem dotykových zařízení, obzvláště pak telefonů a tabletů. Přechod ze starých tlačítkových telefonů nebyl pro uživatele velkým problémem, protože ovládací prostředí se opět posunulo směrem, které je mu více přirozené. To dokazuje i existence notebooků, které mají dotykovou vrstvu. Nyní jsou na trhu zařízení, která umožňují ovládání na základě pohybu. To zní jako další revoluční okamžik, ale zatím nejsou tolik rozšířená a stále se pro ně hledá využití.

Je zřejmé, že každé ze vstupních zařízení má svůj primární účel a těžko ho lze v tomto směru překonat. Velice nepravděpodobně vypadá představa, že by myš v budoucnu úplně nahradila klávesnici, nebo že by myš úplně zanikla v důsledku ještě většího rozšíření dotykové technologie. Klasická vstupní zařízení mají i nadále své místo, ale jsou postupně nahrazována v oblastech, pro které se nehodí.

1.1 Cíl práce

Cílem této práce je seznámit se s pohybovým zařízením Kinect a najít pro něj vhodné využití. Rozsah možností, které toto zařízení umožňuje je velký a bylo by těžké najít takovou situaci, která by využívala všechny funkce naplno. Protože rozpoznávání lidské řeči není v oblasti informačních technologií žádnou novinkou, bude s práce primárně orientovat na využití nového způsobu ovládání, se kterým senzor Kinect přišel. Jde o první verzi tohoto zařízení, a tak se dá předpokládat, že bude mít alespoň drobné chyby. Na konci práce budou zhodnoceny výsledky a navrhnuto možné zlepšení.

2 Přehled řešené problematiky

2.1 Herní pohybové sensory

Herní trh se neustále rozrůstá a tak se výrobci herních konzolí předhánějí, aby přilákali co nejvíce hráčů a také vývojářů her. Není se čemu divit, jenom za první čtvrtletí roku 2014 utratili hráči 983 milionů dolarů za herní hardwarové zařízení. To představuje nárůst o 47 % oproti stejnému období v roce 2013. Tento nárůst je způsoben především uvedení na trh dvou nových verzí konzolí ze současných tří nejsilnějších hráčů – Sony PlayStation, Xbox One a Nintendo Wii U („NPD: Total Industry Consumer Spending on Video Games at $4.6 Billion for Q1 2014", 2014).

Ještě před deseti lety byla všechna zařízení podobná. Odlišovaly se svým výkonem, cenou a měly téměř „na chlup" podobný způsob ovládání. Klasickým představitelem herního ovladače se stal gampad, který výrobci dopilovali k dokonalosti, ale velký prožitek ze hry nabídnout nemohl. Vše se změnilo s příchodem Wii Remote, který jako první přišel s herním pohybovým ovladačem. Tento koncept dokázal přilákat ke hraní her i lidi, kteří jim dosud nevěnovali pozornost (Brain, 2007).

Předchůdci

Chris Schmidt z Architecture Machine Group začal v 70. letech pracovat na projektu s názvem Put-That-There, který se opíral o myšlenku Richarda Bolta. Ve výsledku tento projekt umožňoval rozpoznávání lidské řeči, gest a byl vyvíjen v místnosti o velikosti 4,8 metru × 3,3 metru s velkým projektorem mířícím na zeď. Uživatel seděl v křesle ve vzdálenosti 2,5 metrů před promítací zdí, měl k dispozici magnetickou krychli pro zachycení prostorového vstupu a mikrofon připevněný na hlavě. Díky základním hlasovým pokynům jako „tohle" a „tam" mohl uživatel vytvářet a hýbat se základními tvary po obrazovce. V roce 1980 Richard Bolt navrhnul, že by s mikrofonem nemusel manipulovat uživatel, ale mohl by být zabudovaný přímo do zařízení. Postupným vývojem tohoto projektu bylo dosaženo toho, že uživatel mohl ovládat loď v Karibském zálivu a umisťovat budovy na mapu v Bostonu. Bolt v roce 1993 založil nový projekt, který se jmenoval The Iconic System a byl založen na původním projektu Put-That-There. Nově umožňoval sledování pohybu očí a původní magnetická krychle byla nahrazena speciální rukavicí, která usnadňovala snímání gest ruky. Promítané prostředí bylo přeměněno z původního dvojrozměrného obrazu na trojrozměrný. V 90. letech Mark

Lucente představil uživatelské rozhraní s názvem DreamSpace, které bylo založeno na obrazovém snímání uživatele a na rozdíl od svých předchůdců žádný vstupní ovladač. Uživatelské rozhraní bylo určeno pro IBM Research, které bylo spustitelné na řadě platforem včetně Windows NT. Systém byl prezentován jako schopná alternativa ke standartní myši a klávesnici, které se ale nerozšířila (Webb & Ashley, 2012).

Přirozené uživatelské rozhraní

S nástupem pohybových senzorů, dotykových zařízeních a systémů rozpoznávajících lidskou řeč, se začalo rozvíjet takzvané přirozené uživatelské rozhraní (Natural User Interface). Od klasického uživatelského rozhraní (User Interface) se liší tím, že není ovládáno myší a klávesnicí, ale využívá dotyků, gest a hlasu (Rouse, 2011).

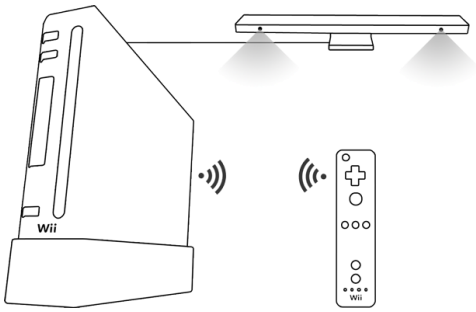
Uživatel je tedy více zapojen do činnosti a aplikace mohou být interaktivnější. To staví návrháře rozhraní do složité situace. Přirozené uživatelské rozhraní by mělo vycházet z běžných pohybů, které dělá člověk každý den. Ideální rozhraní je takové, které dokáže uživatel ovládat bez jakýchkoliv informací o něm, ale protože lidé mají individuální návyky a rozdílnou kulturu, je poměrně těžké přizpůsobit návrh všem. V konečně fázi vychází rozhraní z předem nadefinovaných gest a jiných ovládacích prvků, které musí uživatel znát. Proto se někteří odborníci domnívají, že přirozené uživatelské rozhraní nelze označovat za přirozené (Malizia, & Bellucci, 2012)

Při vstupu nové technologie na trh, která využívá přirozeného uživatelského rozhraní, bývá často srovnávána s filmem Minority Report z roku 2002, který obsahoval v té době velice futuristické ovládání (Webb & Ashley, 2012).

2.1.1 Wii Remote

Wii Remote je herní ovladač, který byl průkopníkem v moderních herních pohybových senzorech (Obrázek 1). Společnost Nintendo v roce 2005 jako první představila ovladač, který je založen především na pohybu a nezakládá se pouze na stisku tlačítek, jako tomu je u klasických gampadů. V době jeho vydání neexistovalo žádné podobné zařízení. Wii Remote je doplňkem pro herní konzoli Wii, která díky tomuto ovladači slavila obrovský úspěch. Konzole sama o sobě není výjimečná, z pohledu počtu nabízených her i z technické hlediska zaostávala za svými konkurenty, ale právě kvůli unikátnímu ovládání se dostala na první příčku v prodejích. Byla tak oblíbená, že v roce 2009 se konzole Wii prodalo více než jejích dvou konkurentů dohromady, tedy Xboxu 360 a PlayStationu 3 (Brain, 2007).

Obrázek 1 Wii a Wii Remote („Amazon.com: Wii: Nintendo Wii: Wii Hardware: Video Games", 2005)



Obrázek 2 Schéma propojení Wii a Wii Remote

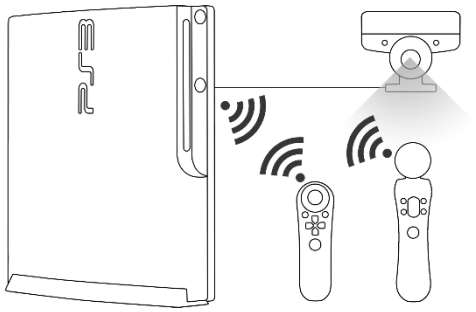
Technologie

Princip použité technologie je poměrně jednoduchý (Obrázek 2). Ovladač obsahuje tříosý akcelerometr, který získává informace pohybu ovladače ve 3D prostoru a zároveň náklon v obou směrech. Pro zlepšení přesnosti slouží takzvaný Sensor Bar, který je umístěn před televizí a obsahuje 5 infračervených LED diod na každé straně. Na přední straně ovladače je umístěna kamera s infračerveným filtrem, která vyhodnocuje směr ovladače na základě viditelnosti a umístění infračervených teček ze Sensor Baru v jejím zorném poli. Všechna tato data se spolu s ostatními předávají konzoli pomocí bezdrátové technologie Bluetooth. Ovladač obsahuje 12 tlačítek, reproduktor, mikrofon, zařízení pro vibrace a rozšiřující port. Ke klasickému ovladači lze připojit nejrůznější doplňky rozšiřující možnosti a prožitek z hraní her (Lee, 2008).

2.1.2 PlayStation Move

PlayStation Move se začal prodávat 15. září 2010, nejdříve v Evropě a o pár dní později v Severní Americe a Japonsku. Vyvinula ho společnost Sony Computer Entertainment a byl určen pro herní konzoli PlayStation 3. PlayStation Move se skládá z kamery PlayStation Eye, pohybového ovladače Motion controller a navigačního ovladače Navigation controller (Obrázek 3). Navigační ovladač kopíruje hlavní prvky ze standardního herního gamepadu, a proto je spíše doplňkem, který má usnadnit pohyb v menu a hodí se pouze pro jistý typ her. Oficiální zpráva z konce roku 2012 udává, že Sony prodalo celkem 15 milionů kusů tohoto herního zařízení („PlayStation®Move Setup", 2013; „PLAYSTATION®MOVE MOTION CONTROLLER TO HIT WORLDWIDE MARKET STARTING THIS SEPTEMBER", 2010).

Obrázek 3 PlayStation Move („PlayStation®Move - PlayStation®3”, 2014)



Obrázek 4 Schéma propojení PlayStationu Move

Kamera

Kamera je důležitou součástí celého systému, protože umožňuje výpočet pozice ovladače ve 3D prostoru (Obrázek 4). Pozorovací úhel kamery se mění v závislosti na jejím nastavení, v úzkém režimu nastaven na 56 stupňů, v širokém režimu na 75 stupňů. PlayStation Eye dokáže snímat obraz v rozlišení 640 pixelů × 480 pixelů při obnovovací frekvenci 60 Hz. Pokud chtějí vývojáři zlepšit odezvu a zachytit plynulejší pohyb, mohou snížit rozlišení na 320 pixelů × 240 pixelů a získat dvojnásobnou obnovovací frekvenci. Kamera také obsahuje matici mikrofonů pro určení zdroje zvuku ve 3D prostoru a schopnosti potlačení šumu („THE TECH BEHIND PLAYSTATION MOVE”, 2010).

Pohybový ovladač

Ovladač je připojen ke konzoli pomocí bezdrátové technologie Bluetooth. Obsahuje 8 tlačítek a na jednom konci má umístěnou svítící kouli, která dokáže zářit všemi dostupnými kombinacemi RGB barev. Díky ní dochází k výpočtu určení polohy ovladače v prostoru. Čím menší svítící kouli kamera zaznamená, tím je ovladač dále od kamery. V případě, že hraje více hráčů najednou, jejich ovladače jsou odlišeny barvou a tak je rozpoznává i kamera. Doplňkově je svítící koule používána k dokreslování efektu ze hry (při střelbě může změnit barvu v červenou a simulovat tak výstřel). V ovladači je umístěn tříosý lineární akcelerometr, tříosý gyroskop a magnetometr. Všichni společně pomáhají určit pohyb a rotaci ovladače. Rapidně tím zvyšují přesnost a plně se na ně spoléhá v okamžiku, kdy je svítící koule ovladače v zákrytu a kamera ji nevidí („THE TECH BEHIND PLAYSTATION MOVE”, 2010; Kališ, 2010; Košťál, & Sláma, 2010).

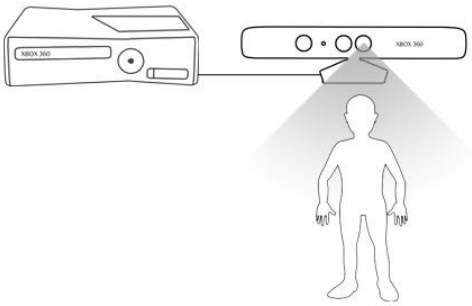
Nabíjecí stanice

Díky tomu, že jsou oba ovladače bezdrátové, je nutné je nabíjet pomocí konektoru mini USB (při nabíjení lze ovladač používat), nebo pomocí nabíjecí stanice, ve které je možné nabíjet ovladače dva (Kališ, 2010).

2.1.3 Microsoft Kinect

Microsoft Kinect, původně s krycím jménem Project Natal, je pohybové zařízení, které bylo vyvinuto pro herní konzoli Xbox 360 za účelem hraní her (Obrázek 5). Po dvou letech od jeho vydání začal Microsoft prodávat verzi připojitelnou k počítači s názvem Kinect for Windows a značně se tak rozšířila využitelnost zařízení (více popsáno v podkapitole Využití). Od svých konkurentů se výrazně liší tím, že nemá žádný ovladač a zavedl tak nový styl ovládání her (Obrázek 6). Sleduje lidskou postavu a pomocí gest a pohybů je možné jej ovládat, jediným ovladačem je tedy lidské tělo (Jana, 2012).

Obrázek 5 Xbox 360 a Kinect (Ruda, 2011)



Obrázek 6 Schéma propojení Xboxu 360 a Kinectu

Veřejnosti byl poprvé představen 1. června 2009 a začal se prodávat 4. listopadu o rok později. Ihned se zapsal do knihy světových rekordů jako nejrychleji se prodávající zařízení spotřební elektroniky. Oficiální stránka Guinessovi knihy rekordů uvádí, že Kinect za prvních šedesát dní (od 4. listopadu do 3. ledna) prodával v průměru 133 333 kusů za den, celkem tedy 8 milionů kusů. Do konce roku 2013 Microsoft prodal více než 24 milionů kusů herního zařízení (Catuhe, 2012; „Fastest-selling gaming peripheral", 2011; „Microsoft says Xbox 360 sales have surpassed 76 million units, Kinect sales top 24 million", 2013).

Výroba Kinectu pro Windows bude v roce 2015 ukončena, protože ho nahradila nová verze Kinect for Windows v2, která se začala prodávat v říjnu 2014. Důvodem stálého prodeje starší verze je ten, že některé firmy mohou mít dlouhodobé závazky vůči původní verzi. Díky novému vývojovému balíčku je potřeba přizpůsobit dosavadní aplikace novému senzoru, a proto je přechod pozvolný („Original Kinect for Windows sensor sales to end in 2015", 2014).

Historie

Když začalo Nintendo prodávat svůj herní pohybový ovladače Wii Remote, společnost Microsoft se rozhodla, že začne pracovat na projektu, který by Wii Remote překonal.

Peter Moore (tehdejší šéf divize pro Xbox) vytvořil dva na sobě nezávislé týmy, jeden pracoval s technologií od společnosti PrimeSense a druhý s technologií, která vyvinula firma 3DV. Původní plán očekával, že dojde k představení výsledků už ve třetím kvadrantu roku 2007, tento plán ale nebyl dodržen. Prototyp zařízení vzniknul až v roce 2008 a to od týmu, který pracoval s technologií PrimeSense. Obsahoval RGB kameru, infračervený zářič a infračervený senzor, v podstatě se nijak významně neodlišoval od prodejní verze. Významně se ale odlišoval od podobných zařízení, které také dokázaly měřit hloubku obrazu. Ostatní zařízení využívaly metodu „Doba letu“, která pracovala na základě měření času mezi vysláním paprsku ze zařízení, odražením se od objektu před ním a jeho návratem. Metoda, která se v Kinectu využívá, bude detailněji popsána v kapitole 2.2.2. Zajímavostí je, že i přesto, že technologie od společnosti 3DV nebyla ve finále použita, Microsoft podnik v roce 2009 koupil za 35 milionů dolarů, aby tak zabránil případným budoucím patentovým sporům. Když byl prototyp v roce 2008 představen vedení společnosti, byl schválen a dostal pracovní označení Project Natal. Po hardwarové stránce bylo zařízení v podstatě hotové, musel se ale vyvinout software, který by byl schopný data využít, rozpoznat člověka stojícího před senzorem a identifikovat jednotlivé části jeho těla. Do zařízení byl přidán mikrofon a tím vznikla potřeba naučit Project Natal porozumět lidské řeči. Tým projektu se obrátil s pomocí na odborníky ze společnosti Microsoft Research, což byla nezvyklá věc, protože Microsoft Research se běžně zabývá vývojem v počítačové vědě, nikoli komerčními produkty. Naslouchání lidské řeči nebylo pro společnost Microsoft žádnou novinkou, jelikož tuto funkci používal od roku 2006 ve svém operačním systému Windows Vista (Webb & Ashley, 2012).

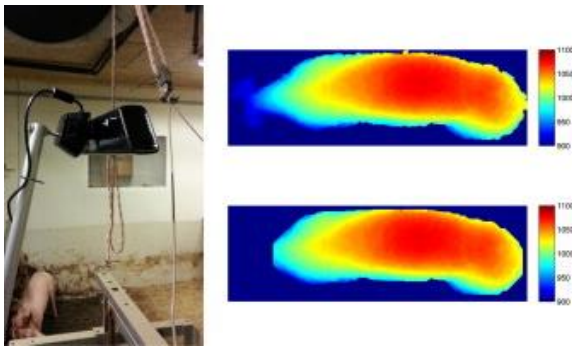
Srovnání Kinectu pro Windows a pro Xbox

Obě zařízení vypadají a chovají se velice podobně. Z pohledu uživatele lze jen těžko hledat rozdíly. Z pohledu vývojáře se ale v několika ohledech liší. Původně byl Kinect vyvinut jako herní zařízení a doplněk pro herní konzoli Xbox a tudíž se nepředpokládalo, že by měl uživatel pomocí něj ovládat nějaké aplikace. Proto vzniknul Kinect for Windows, který má lehce upravený firmware a je možné u něj přepínat mezi standartním a blízkým módem, které upravují ideální vzdálenost uživatele od senzoru. Pro verzi určenou k herní konzoli Xbox musí být hráč vzdálen minimálně 80 centimetrů, verze určená pro Windows umožňuje sledování již od vzdálenosti 40 centimetrů. Druhou odlišností je licenční smlouva, která ukládá, že pouze Kinect for Windows je určený pro komerční aplikace. Poslední malý rozdíl je v USB kabelu, který by měl být u počítačové

verze kvalitnější a robustnější, protože se u něj předpokládá častější namáhání (Jana, 2012).

Využití

Kinect se díky svému sledování postavy člověka, rozpoznávání pohybů a gest, měření vzdáleností jednotlivých objektů a porozumění lidské řeči, dá využít v řadě oborů. Ve zdravotnictví našel uplatnění pro cvičení, rehabilitace, sledování pacientů, Microsoft Research dokonce vyvinul systém „Touchless Interaction in Medical Imaging“, který se využívá při operacích, kde operující doktor může snadno pomocí gest ovládat 3D model operované části těla a získat o něm bližší informace (Obrázek 8). Senzor je také využíván pro bezpečnostní případy, lehce totiž dokáže odhalit pohyb člověka na daném území. Neuvěřitelně ale zní informace, že zařízení pro tento účel využívá i Jižní Korea, aby kontrolovala hranice se Severní Koreou. Senzor další využití nalezl na farmách při měření váhy prasat, na základě hloubkového senzoru a analýzy dat dokáže určit váhu zvířete s odchylkou 4 - 5 % (Obrázek 7). Dále se senzor používá při vzdělávání (forma škola hrou), jako trenér (existuje i výukový simulátor pro policisty), robotice, virtuální realitě a v dalších oborech (Jana, 2012; „Kinect Launches a Surgical Revolution”, 2012; „Microsoft's Kinect guards South Korea's borders against trespassers”, 2014; Kongsro, 2014; „Microsoft's Kinect guards South Korea's borders against trespassers”, 2014).



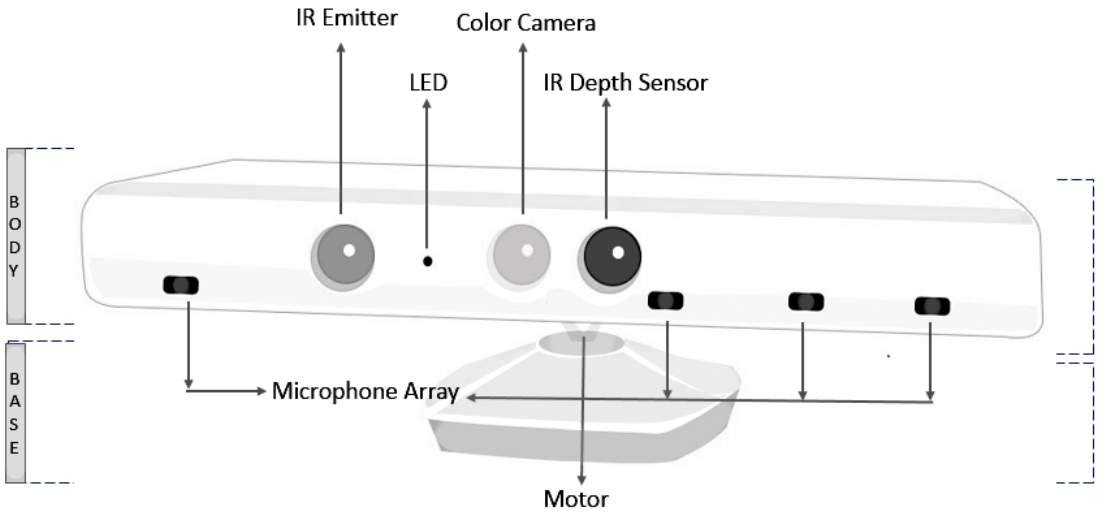
Obrázek 7 Měření váhy prasat (Kongsro, 2014)

Obrázek 8 Touchless Interaction in Medical Imaging („Kinect Launches a Surgical Revolution”, 2012)

2.2 Technologie senzoru Kinect

2.2.1 Komponenty

Herní zařízení Kinect je horizontální zařízení, které obsahuje kamery, infračervený zářič a matici mikrofonů (Obrázek 9). Vše je zabaleno do malého černého boxu a připojeno k malému motůrku v základně zařízení umožňující změnu náklonu. Zařízení se připojuje pomocí USB kabelu a je napájeno pomocí externího adaptéru (Jana, 2012).



Obrázek 9 Komponenty senzoru (Jana, 2012)

RGB kamera

Kamera dokáže zachytit obraz (v RGB[1], nebo YUV[2] formátu) v rozlišení 1280 pixelů × 960 pixelů při 12 snímcích za sekundu, V případě potřeby zlepšení odezvy, lze snížit rozlišení na polovinu (640 pixelů × 480 pixelů) a při 30 snímcích za sekundu. Omezení velikosti rozlišení je dáno především kvůli propojení přes USB konektor 2.0, proto se využívá komprese a dekomprese, aby byl přenos dostatečně rychlý. Kamera dokáže automaticky vyvažovat bílou, saturovat barvy a vyhýbat se jejich třepotání. Na rozdíl od klasických webových kamer, nedochází k filtrování infračerveného světla, které je poté využíváno pro měření hloubky obrazu. Zorné pole kamery je 57,5 stupňů horizontálně a 43,5 stupně vertikálně, které je možné upravovat díky motůrku v základně senzoru. Data

[1] Barevný model založený na smíchání červené, zelené a modré barvy
[2] Barevný model založený tříprvkovém vektoru [Y, U, V], kde Y je jasová složka, U a V jsou barevné složky

z této kamery jsou nakonec využívána pro zjišťování detailů lidí a objektů (Webb & Ashley, 2012; Giorio & Fascinari, 2013).

Infračervený zářič

Zářič neustále ozařuje objekty před sebou infračervenými paprsky, které jsou normálně pro lidské oko neviditelné, aby tak spolu s hloubkovým senzorem určil jejich vzdálenost od senzoru. Vysílá pseudonáhodný vzor o celkové velikosti 633 teček × 495 teček, který je rozdělen na subvzorky o velikosti 3 tečky × 3 tečky. V momentě dopadu paprsků na objekt se subvzorek vykreslí a dojde k vytvoření základu pro určení hloubky obrazu (Giorio & Fascinari, 2013).

Hloubkový senzor

Hloubkový senzor obsahuje infračervenou kameru, která dokáže přečíst vysílané tečky z infračerveného zářiče a přeměnit je na informaci o vzdálenosti jednotlivých objektů. Vytváří proud jednotlivých 3D snímků, které mohou být v rozlišení 640 pixelů × 480 pixelů, 320 pixelů × 240 pixelů, nebo 80 pixelů × 60 pixelů. Zorné pole infračervené kamery je totožné s RGB kamerou (Jana, 2012).

Mikrofon

Zařízení obsahuje celkem čtyři mikrofony, které jsou umístěny v zadní části zařízení a obsahují 24bitový převaděč do digitálního signálu. Zaznamenávaný zvuk je kódován pomocní pulzně kódové modulace[3] se vzorkovou frekvencí 16 KHz a rozlišením 16 bitů. Spolupráce mikrofonů dokáže výrazně potlačit hluk, detekovat ozvěnu a určit místo a směr odkud zvuk vychází (Giorio & Fascinari, 2013)

Motůrek a akcelerometr

Motůrek slouží k úpravě vertikálního úhlu náklonu senzoru. Zorné pole ve vertikálním směru je 43 °, motůrek je schopný otočit s Kinectem o 27 ° oběma směry, pokud je to možné. Zajímavé totiž je, že úhel nastavení náklonu není počítán proti základně, ale proti vodorovině. To tedy znamená, že při nastavení úhlu 0 ° bude kamera vždy nasměrována přímo před sebe. Vývojář má k dispozici informaci, jestli se nějaký z hráčů přiblížil hranici zorného pole, může tak během hry upravovat úhel náklonu a zajistit správný pozorovací úhel (Jana, 2012)

[3] Metoda převodu analogového zvukového signálu na signál digitální

Led dioda

Její účel je prostý – indikace stavu zařízení. V případě, že svítí zeleně, připojení Kinectu k počítači proběhlo bez problému. Nic to ale nevypovídá o stavu externího napájení, které je pro chod zařízení nezbytné. Červená barva led diody značí, že senzor nepracuje tak jak má (Giorio & Fascinari, 2013)

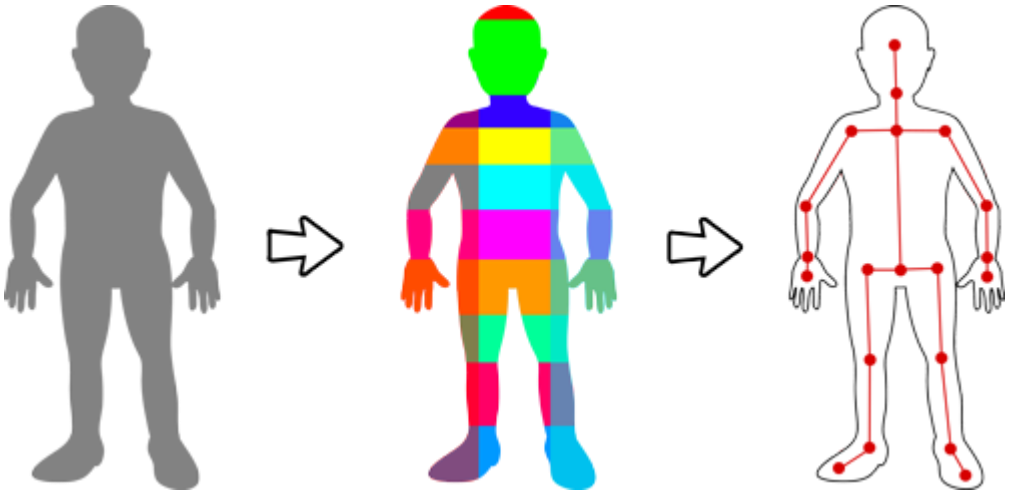
2.2.2 Sledování postavy

Na základě kombinace infračerveného zářiče a infračervené kamery je Kinect schopný zjišťovat hloubku obrazu v jeho zorném poli. Měření vychází z konečné velikosti subvzorku promítnutého na předmět v zorném poli. Čím menší velikost subvzorku infračervená kamera zaregistruje, tím je předmět, který je tímto subvzorkem ozařován, dále od senzoru (Obrázek 10). Celkově tato dvojce takto snímá 34 815 bodů, čímž získává poměrně detailní 3D obraz (Borenstein, 2012).

Obrázek 10 Měření hloubky obrazu senzorem Kinect (Flatley, 2010)

Z těchto dat potom vychází algoritmus, který odfiltruje okolí a pracuje pouze s tou částí obrazu, na které je zachycen člověk. Tento proces je poměrně náročný, protože musí zohledňovat celou řadu odlišností (různou šířku a výšku postavy, brát v potaz oblečení a tak dále). Výpočet pozice postavy je založen na porovnávání s referenčními daty, které zahrnují nejrůznější kombinace postav a dalších rozdílů. Výsledkem toho procesu je pouze silueta (Obrázek 11 vlevo), která by bez aplikace dalších algoritmů byla považována za pouhý objekt jako jakýkoliv jiný. K určení jednotlivých částí těla se silueta rozkouskuje na malé části pomocí rozhodovacích stromů[4] – každý pixel jimi prochází a přiřazuje se k některé z částí postavy. Silueta se tedy rozdělí na několik segmentů (Obrázek 11 uprostřed) a na základě výpočtu pravděpodobnosti dochází k určení pozic jednotlivých bodů (Obrázek 11 vpravo). Výsledná pozice je určena třemi rozměry, dva jsou použity pro určení umístění na obrazu a třetí pro vzdálenost od senzoru (Jana, 2012).

[4] Data miningová technika pro získávání netriviálních a skrytých dat



Obrázek 11 Proces rozpoznání částí lidské těla

2.3 Vývojářské nástroje a ostatní technologie

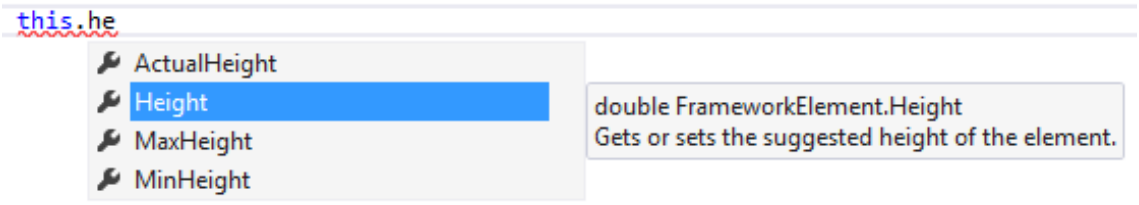
2.3.1 Microsoft Visual Studio

Společnost Microsoft vypustila první verzi Visual Studia v roce 1995, který byl nástupcem tří vývojových prostředí (Microsoft Visual Basic, Visual C++ a Visual FoxPro), které sjednotil v jedno. Velký úspěch zaznamenala zejména verze s označením 6.0, která se těšila velké oblibě programátorů do té doby, než se v roce 2002 začala společnost Microsoft orientovat na .NET Framework. V současné době nabízí vývoj v několika jazycích a nejnovější verzí je Visual Studio 2013 ve čtyřech edicích – Express, Professional, Premium a Ultimate (Desjardins, 2014).

Intellisense

Jedním z důvodů, proč je Visual Studio tak oblíbené, je ten, že nabízí funkci Intellisense. Ta umožňuje okamžitou zpětnou vazbu o chybách v kódu, která se projeví červeným vlnkovitým podtržením v místě, kde se chyba nachází. V některých případech dokáže i navrhnout řešení – například pokud je používán nedefinovaný datový typ, Intellisense dokáže tento typ, nebo třídu vygenerovat. Také v momentě, kdy vývojář začne psát kód, Intellisense napovídá seznamem platných proměnných, typů, tříd, metod a dalších členů, které je možné použít. Snižuje tedy riziko chyby kódu ještě předtím, než se ve skutečnosti stane. Programátor si tedy nemusí pamatovat přesné názvy členů, protože mu jsou neustále napovídány (Obrázek 12). Rapidně zvyšuje efektivitu práce, dvojnásob v případě, kdy je programovací jazyk citlivý na velikost písmen (takzvaně case sensitive),

a vývojář se tak může soustředit především na samotný algoritmus a implementaci (Johnson, 2014).



Obrázek 12 Ukázka nápovědy použitelných členů od Intellisense

2.3.2 Windows Presentation Foundation

WPF je moderní grafický systém, který je určen především pro operační systémy Windows Vista, Windows 7 a Windows 8 (v desktop módu). Oproti předešlému způsobu vývoje aplikací jde o radikální změnu, která nabízí inovativní funkce jako například vestavěnou hardwarovou akceleraci nebo nezávislost vzhledu na rozlišení obrazovky (MacDonald, 2012).

Historie

Operační systém Windows zpřístupňoval řadu funkcí, které mohly aplikace využívat pro svoje rozhraní. Tento systém byl původně navržen pro vývoj v jazycích C nebo C++ a byl to přímý způsob, jak s operačním systémem Windows komunikovat. Funkce se týkaly zejména přístupu k paměti a souborovému systému, síťových služeb, přístup k Windows Shell obsahující celkové uživatelské rozhraní Windows (včetně plochy, hlavního panelu a Windows Exploreru), GDI (funkce pro výstup grafiky na obrazovky, tiskárny a jiná zařízení) a dalších funkcí (Andrade & Livermore & Meyers & Van Vliet, 2007).

Syntaxe a architektura

Technologii využívá XAML[5], která je na klientské straně. Vývoj WPF aplikací je nejsnazší ve vývojovém prostředí Visual Studio (zaměřené na vývojáře) nebo Expression Blend (zaměřené především na designéry). Celá myšlenka je založena na oddělení uživatelského rozhraní od samotného kódu. Uživatelské rozhraní je definováno v XAML souboru obsahující strukturu a jednotlivé prvky, zatímco kód je umístěn v jiném souboru, který definuje chování ovládacích prvků. (Garofalo, 2011).

[5] značkovací jazyk využívaný k popisu grafického rozhraní

2.3.3 Microsoft SQL server

Microsoft SQL Server je databázový systém, který dokáže přehledně ukládat velké množství informací. Současným trendem je ukládání téměř všech získaných informací, objem uložených dat se za posledních pět let vyšplhal na desetinásobek a navíc roste exponenciálně. Možnost uchování dat je stejně důležitá, jako rychlý přístup k těmto datům, proto se v poslední době klade stále větší důraz na dobu zpracování dotazů (Sarkar, 2013).

Současná verze 2014 je založena na úspěchu verze minulé. Nabízí uživatelům možnost bezpečného uložení a ochranu svých informací, rychlý přístup k nim a nástroje pro tvorbu a správu nejnáročnějších databázových aplikacích (Mistry & Misner, 2014).

Alternativní databázové systémy

Na trhu existuje celá řada alternativních databázových systémů. Dokonce i sám Microsoft vyvíjí jednoho konkurenta – Microsoft Access. Úspěch Accessu je dán především tím, že je součástí některých verzí populárního kancelářského balíčku Microsoft Office. V praxi se využívá hlavně pro malé a domácí projekty, kvůli tomu, že systém není dostatečně rychlý, škálovatelný a flexibilní. Největším rivalem v této kategorii je zřejmě Oracle, o kterém se mluví jako o „jedničce v oboru“. Je totiž dostatečně robustní pro velké projekty a nabízí velké množství nástrojů. Tento systém se od Microsoft SQL Serveru výrazně odlišuje svojí prodejní politikou. Společnost Microsoft se snaží prodat licenci, která obsahuje téměř vše, jednotlivé balíčky nelze dokupovat (pouze upgradovat na lepší edici), zatímco společnost Oracle prodává základní databázový systém a postupně je nutné dokupovat jednotlivé funkce. Velké rozdíly jsou vidět i při instalaci, Microsoft SQL Server je mnohem uživatelsky přívětivější a jednodušší. Největší nevýhoda oproti konkurenci je ta, že na serveru musí být nainstalovaný .NET Framework, a to i v případě, že není využíván. Téměř všechny programovací jazyky jednotlivých databázových systémů jsou založeny na standardu ANSI-92, základ je tedy pro všechny stejný, liší se především v syntaxi a v implementaci funkcí specifické pro jednotlivé systémy. Programátoři tak při změně databázového systému nemívají velké problémy (Dewson, 2012).

2.3.4 .NET Framework

Před vznikem .NET platformy používali programátoři k vývoji aplikací pro operační systém Windows model objektových komponent (COM). Ten umožňoval vývoj

knihoven[6], které bylo možné využívat napříč různými programovacími jazyky. To bylo jistě velmi přínosné, ale model nebyl příliš rozvinutý a měl komplikovanou strukturu, Dnes se od využívání tohoto modelu upustilo a do značné míry byl nahrazen platformou .NET. Tu Microsoft poprvé představil v červnu 2000 ve verzi 1.0. Od té doby vzniklo šest dalších verzí, přičemž aktuální verze nese označení 4.5 (Thai & Lam, 2002).

Platforma se celkově skládá ze tří bloků: Common Language Runtime (CLR), Common Type System (CTS), Common Language Specification (CLS). V hierarchii je CLR ve spodní vrstvě a jeho úkolem je spouštět aplikace, poskytovat jim služby a starat se úkoly na nižší úrovni (správa paměti, zajišťování bezpečnosti, hostování aplikací a řízení vláken). CTS popisuje jednotlivé datové typy a struktury, komunikaci mezi nimi a jakým způsobem jsou tato data reprezentována. Na vrcholu pyramidy je Common Language Specification, který definuje běžné typy a struktury, které je možné používat ve všech programovacích jazycích platformy .NET. To umožňuje převod zdrojového do jiného .NET jazyka nebo nástavbu aplikace v jiném .NET jazyku, než je originál (Troelsen, 2012).

2.3.5 Kinect Studio

Jako pomocný nástroj při vývoji aplikací vzniklo Kinect Studio, které dokáže nahrávat a přehrávat data ze všech senzorů. Tato vstupní data lze uložit do souboru s příponou .xed a poté je opětovně použít. Vývojáři ho oceňují zejména při testování, protože v tomto případě by bylo těžké zopakovat před Kinectem naprosto stejný pohyb jako předtím, aby se zjistilo, jestli je chyba skutečně opravená. Umožňuje tak vývojáři se plně soustředit na správnou implementaci a pozorovat zdrojový kód za běhu aplikace (Catuhe, 2012).

2.3.6 Software Development Kit (SDK)

Software Development Kit je balíček knihoven, které usnadňují programátorovi komunikaci a získávání dat od jednotlivých komponent senzoru. Kromě toho umožňuje rozpoznávání gest a hlasového vstupu, sledování lidské postavy a určování jejích jednotlivých části, získávání informací z akcelerometru a spoustu dalších funkcí. Programátor tedy používá již hotové metody pro ovládání samotného zařízení a získávání

[6] označení pro soubor funkcí a procedur, který může být sdílen více počítačovými programy

výstupů z něj, implementaci rozhraní senzoru zajišťuje Software Development Kit (Webb & Ashley, 2012).

Kinect for Windows SDK

Oficiální knihovny nabízí společnost Microsoft zdarma ke stažení na svých webových stránkách. První beta verze byla vydána v červnu 2011 a od té doby SDK postupně prodělalo celou řadu změn. Nejnovější verze 1.8 zahrnuje kromě klasických knihoven i další, které se netýkají přímo zařízení, přesto však souvisí s vývojem aplikací pro Kinect. Jedná se o ovládací prvky, kterou jsou pro používání v kombinaci se senzorem připravení. (Jana, 2012).

Kinect for Windows SDK lze spustit pouze na operačním systému Windows 7 nebo 8, popřípadě 8.1. Počítač musí mít minimálně dvoujádrový a 32bitový nebo 64bitový procesor s frekvencí 2,66 GHz, 2 GiB operační paměti, grafickou kartu, která podporuje DirectX 9.0c, a port USB 2.0 (Webb & Ashley, 2012).

Neoficiální SDK

Díky značné popularitě Kinectu vznikly i neoficiální balíčky knihoven, které jsou vyvíjeny nadšenci pro toto zařízení. Jeden z mnoha důvodů, proč takové projekty vůbec vznikly, je možnost vyvíjení aplikací i na jiné operační systémy – Mac OS X nebo Ubuntu. Jako zástupce alternativních SDK lze uvést libfreenect od komunity OpenKinect a OpenNI od PrimeSense. Odlišují se zejména účelem, pro které byly navrženy. Některé se zaměřují více na příkazy na nižší úrovni (ovládání jednotlivých částí, přístup k surovým datům), jiné především na přirozené uživatelské rozhraní. Většina z těchto balíčků je volně dostupných i v podobě zdrojového kódu (takzvaný open source) a dokonce je možné je používat i pro komerční účely (St. Jean, 2012).

2.3.7 Face recognizing

V poslední době se začíná rozšiřovat ověřování na základě biometrických údajů. Klasické přístupy založené na heslu totiž nejsou a ani nemohou být dostatečně bezpečné. Nejslabším a nejzranitelnějším místem je uživatel, který si silné heslo jen stěží zapamatuje a je dost velká pravděpodobnost, že ho i zapomene. Při volbě slabého hesla může být prolomeno, nebo uhádnuto. Podobná rizika nesou i čipové karty, které mohou být ukradeny, ztraceny nebo zničeny. Biometrické údaje ukrást nelze, ale také může dojít k situacím, které povedou k neúspěšné autorizaci. Při poškození tkáně na prstu může dojít

k neshodě otisku, ověřování na základě hlasu může mít problémy s okolním šumem. Zařízení ověřující identifikaci jsou často poměrně drahá, vyžadují zafixování postavy v určité pozici a bývají náchylná na sebemenší pohyb. Face recognizing je ale v tomto směru trochu jiný. Rozpoznání obličeje může být provedeno i z fotografie pořízené v pohybu, lze ho provést na základě jedné nebo více kamer a je bezdotykové, čímž se snižuje riziko přenosu bakterií (Jafri & Arabnia, 2009).

Hlavní problém v rozpoznání obličeje se skrývá v ověření vstupních dat (nejčastěji ve formě fotografie) a databází obličejů známých osob. Způsob, jakým je rozpoznání prováděno, lze rozdělit na holistické metody, metody založené na rysech a hybridní metody. Jednou z nejznámějších holistických metod je metoda Eigenface, která na základě trénovacího setu obrázku vytvoří vzory a na základě nich probíhá porovnávání. Metody založené na rysech se orientují na umístění a tvary důležitých částí obličeje (oči, nos, ústa). Hybridní metody jsou založeny na kombinaci dvou předešlých kategorií (Parmar, & Mehta, 1970).

3 Metodika

Pro práci se senzorem Kinect jsem se rozhodl využít oficiální vývojový balíček Kinect for Windows SDK 1.8. Tato volba dala velký předpoklad pro výběr dalších nástrojů. Oficiální knihovny lze využívat pouze ve spojení s operačním systémem Windows a zároveň jsou určeny pro vývoj aplikací v .NET jazycích. Vývojářské nástroje ani operační systém od společnosti Microsoft nebývají zadarmo, některé pouze v odlehčených verzích, ale díky spolupráci společnosti Microsoft s Jihočeskou univerzitou mám jako student přístup k většině jeho produktů bez poplatku.

Pro vývoj aplikace jsem tedy zvolil programovací jazyk C# ve vývojovém prostředí Microsoft Visual Studio Premium 2013. Tento jazyk má velkou základnu fanoušků díky své jednoduchosti a ve spojení s Visual Studiem nabízí rychlý a kvalitní vývoj. Visual Studio je přehledné a nabízí spoustu užitečných funkcí popsaných v kapitole 2.3.1. Pro použití tohoto nástroje je nutné jej pouze nainstalovat, žádná jiná nutná nastavení nevyžaduje.

Visual Studio obsahuje knihovny, které umožňují snadné připojení k Microsoft SQL Serveru. Spolupráce je natolik provázaná, že obsahuje i grafické rozhraní pro správu tohoto databázového serveru. Pro přístup aplikací k Microsoft SQL Serveru je nutné při instalaci zadat heslo k výchozímu uživatelskému jménu „sa“ a po dokončení instalace přepnout způsob autentizace z původní hodnoty „Windows Authentication mode“ na hodnotu „SQL Server and Windows Authentication mode“. Nastavení uživatelského jména a hesla je možné udělat i po instalaci v sekci „Security“, kde se nachází správa přístupů i s jejími právy.

Pro využívání vývojového balíčku je nutné ve Visual Studiu přidat do projektu reference na knihovny pro práci se senzorem Kinect, které budou aplikace využívat. Tyto knihovny se nachází ve složce, která byla uvedena při jejich instalaci a podadresáři Developer Toolkit v1.8.0\bin. Kromě přidání referencí je nutné knihovny zkopírovat do výstupní složky, kde se nachází zkompilovaná a spustitelná aplikace, a toto umístění vždy zachovat.

Při vývoji je nutné použít i některé z klasických ovládacích prvků:

- User control – skupina ovládacích prvků, které jsou sdružené za účelem vytvoření nového ovládacího prvku

- Grid – mřížka definující uspořádání, zarovnání a případně velikost ovládacích prvků
- ListView – ovládací prvek sloužící pro zobrazení množiny prvků v různém grafickém rozvržení
- Label – popisek
- Button – tlačítko
- WrapPanel – panel řadící svoje potomky ve vertikálním či horizontálním směru

Implementace metod rozpoznávajících obličej popsaných v kapitole 2.3.7 je náročná věc a překračovala by rámec této práce. Proto jsem přistoupil k jednoduchému porovnávání na základě relativních vzdáleností jednotlivých částí obličeje a určité odchylky. Relativní vzdáleností je myšlena velikost jedné části obličeje v poměru k velikosti druhé části obličeje. Při pouhém srovnávání jednoduchých vzdáleností se algoritmus vystavuje problému, že uživatel bude pokaždé v jiné vzdálenosti od senzoru, velikost obličeje na snímku bude rozdílná, a proto si hodnoty nebudou odpovídat.

4 Řešení a výsledky

4.1 Popis systému

Systém se snaží vylepšit dosavadní terminály, které dnes běžně fungují na principu skenování čárového kódu a kontroly přístupu. Lístky s čárovými kódy často tisknou zaměstnanci, čímž dochází k nevyužití fyzických i duševních kapacit pracovníka, a každým výtiskem se zvyšují náklady na papír a barvu. Využitím senzoru Kinect je možné, aby si návštěvník sám zvolil, o které služby má zájem, a následná identifikace probíhala na základě rozpoznání jeho obličeje.

Při vstupu do zařízení stiskne návštěvník tlačítko „Vstup“ a vybere si ze seznamu sekcí, o které má zájem. U každé z nich zvolí určitou variantu přístupu, některé přístupy mohou být založeny na časovém omezení, jiné na počtu vstupů a některé bez omezení. Po výběru je spočítána celková cena, kterou musí návštěvník zaplatit. Potvrzením volby dojde k identifikaci obličeje a zapamatování si těchto dat pro pozdější využití. Při výstupu stiskne člověk tlačítko „Výstup“ a proběhne porovnání jeho obličeje s databází obličejů, které byly zaznamenány u vstupu. Při úspěšném spárování je výstup dokončen, v ostatních případech je nutné zkusit rozpoznání obličeje znovu, nebo zavolat obsluhu.

4.2 Popis tříd

4.2.1 Třída cDatabaze

Z důvodu potřeby přístupu k datům z databáze vznikla třída cDatabaze, která tyto požadavky zajišťuje. Jedinou globální proměnnou je proměnná connection, která uchovává informace o připojení k databázovému serveru, žádné další proměnné ani vlastnosti třída neobsahuje. Konstruktor vyžaduje parametr typu string, který definuje název serveru, uživatelské jméno a heslo, jméno databáze, kam je požadovaný přístup, a další volitelné informace. Vytvoření instance připojení zajistí metoda VytvorInstanciPripojeni, která je konstruktorem volána.

Metoda KontrolaPripojeni vrací logickou hodnotu v závislosti na aktuálním stavu připojení. Pokud je připojení otevřené, ihned vrací hodnotu pravda, pokud ne, pokusí se o otevření a podle výsledku vrátí hodnotu pravda, nebo nepravda. Tato metoda se hodí pro otestování připojení a využívají ji všechny další metody.

Pro získání dat z databáze jsou definovány dvě metody. K získání tabulky, nebo její části, slouží metoda DotazRadky, pro použití agregačních funkcí a výpočtu číselné hodnoty je potřeba volat metodu DotazHodnota. Metoda DotazRadky je přetížená a je možné jí předat parametr typu string s SQL dotazem, nebo parametr typu SqlCommand, který umožňuje tvorbu složitějších SQL dotazů. U metody DotazHodnota se nepředpokládají složité dotazy, a proto vyžaduje pouze parametr typu string s SQL dotazem.

Vkládání dat do databáze zajišťuje metoda VlozitRadky, která je přetížená a vyžaduje parametr typu string s definovaným SQL příkazem, nebo parametr typu SqlCommand. Po vložení dat do databáze vrací číselnou hodnotu, která udává, kolik řádků bylo vloženo. V případě, že se k databázi nelze připojit, vrací prázdnou hodnotu (null). Na stejném principu funguje i metoda UpdateRadky, která provádí změnu dat, je rovněž přetížená a vrací informaci o počtu změněných řádků.

Metoda SmazatRadky se stará o smazání řádků z databáze a požaduje parametr typu stringu s definovaným SQL příkazem. Vrací počet řádků, které byly smazány.

4.2.2 Třída cCena

Třída slouží k nadefinování ceníku, který je součástí třídy cSekce (podrobněji rozebrána v následující kapitole 4.2.3). Reprezentuje jednu z možných variant, které jsou k dané sekci k dispozici. Varianta může vycházet z určitého časového nebo početního omezení (proměnná pocetJednotek), má určitou cenu (proměnná cena) a potřebuje stanovit, o jaký typ omezení se jedná (proměnná datovyTyp). Struktura je přizpůsobena pro práci s databází, proto obsahuje proměnné pro uchování informace o identifikátoru v tabulce (proměnná id) a identifikátoru sekce, které je určena (proměnná id_sekce). Vlastnosti s obdobnými názvy zajišťují přístup k těmto proměnným v režimu readonly. Vlastnost CenaToString zajišťuje vypsání ceny s českou měnou, za proměnnou cena totiž přidává řetězec „Kč“, PocetJednotekToString upravuje výpis jednotek ve vhodném formátu na základě hodnoty proměnné datovyTyp. Pokud jde o časový rozsah, vypíše jednotky ve formátu HH:MM, při celočíselné hodnotě přidá za hodnotu krát pro zdůraznění, že jde o počet vstupů, a když je datovyTyp roven „Infinity“, nahradí výpis jednotek řetězcem „Neomezený vstup“.

4.2.3 Třída cSekce

Třída cSekce reprezentuje jednotlivé sekce, které jsou uživatelům k dispozici, žádný jiný účel než uchovávání informací nemá. Struktura je přizpůsobena pro používání

v kombinaci s databází. Má celkem osm privátních proměnných, sedm z nich je základem pro pozdější vytvoření tlačítka reprezentující sekci a poslední proměnná slouží k uchování indexu varianty z listu datového typu cCena, která byla uživatelem vybrána. S těmito proměnnými jsou svázány i vlastnosti, které jsou určené pouze pro čtení a nikdo je tak nemůže samovolně nastavovat. Dále se zde vyskytuje vlastnost vracející ImageSoure pro využití ovládacím prvkem Image, list datového typu cCena, který odráží varianty ceníku pro danou sekci, a vlastnost VybranaVarianta, vracející vybranou variantu z ceníku.

Konstruktor vyžaduje parametr typu DataRow, ten reprezentuje řádek z databáze, a v těle konstruktoru dochází k přiřazení jeho hodnot proměnným.

4.2.4 Třída cOblicej

Struktura této třídy je navržena podle výstupu z procesu rozpoznávání obličeje. Výsledkem je kolekce bodů částí obličeje, z nichž jsem některé vybral pro výpočet sedmi základních poměrů – šířka obličeje s výškou obličeje, výška nosu s výškou obličeje, výška čela s výškou obličeje, výška brady s výškou obličeje, šířka nosu s šířkou obličeje, šířka úst s šířkou obličeje a šířka brady s šířkou obličeje. Pro všechny tyto poměry jsou vytvořeny proměnné s odpovídajícím názvem datového typu double, protože se u nich zcela jasně předpokládá číselná desetinná hodnota. Poslední proměnnou je foto, které reprezentuje fotografii pořízenou při rozpoznávání obličeje. Ke všem proměnným existují vlastnosti s podobným názvem a umožňují pouhé čtení hodnoty, zápis není povolen.

Třída obsahuje jeden konstruktor a jednu metodu. Konstruktor požaduje parametry datového typu EnumIndexableCollection obsahující zmíněné body po rozpoznání obličeje a datového typu byte[] reprezentující pořízenou fotografii. V těle konstruktoru následně dochází k výpočtu poměrných vzdáleností pomocí metody VzdalenostDvouBodu a přiřazení její návratové hodnoty proměnným. Zmíněná metoda požaduje dva parametry typu PointF, které obsahují souřadnice os X a Y. Výpočet je založen na klasickém vzorečku vzdálenosti dvou bodů v rovině.

```
Math.Sqrt(Math.Pow(bod1.X - bod2.X, 2) + Math.Pow(bod1.Y - bod2.Y, 2));
```

Ukázka kódu 1 Výpočet vzdálenosti dvou obličejových bodů

4.3 Databázový model

Navrhnutí databáze vycházelo ze struktury členů použitých tříd a požadavků na uchování informací pro zajištění hladkého chodu systému. Po celkové analýze vznikly čtyři tabulky, jejichž propojení je znázorněno v Příloha C.

4.3.1 Tabulka Sekce

Pro definování jednotlivých sekcí vznikla tabulka, která má 6 sloupců (Obrázek 13). Kromě klasického celočíselného identifikátoru Id, který je i primárním klíčem tabulky, obsahuje sloupce:

- Nazev pro uchování názvu sekce
- Obrazek k upřesnění detailu sekce
- Sirka a Vyska určují velikost tlačítka, které bude ve výpisu sekcí vykresleno
- Poradi pro definování pořadí ve výpisu tlačítek sekcí

Column Name	Data Type	Allow Nulls
Id	int	☐
Nazev	nchar(20)	☐
Obrazek	image	☑
Sirka	smallint	☑
Vyska	smallint	☑
Poradi	smallint	☐

Obrázek 13 Databázový slovník tabulky Sekce

4.3.2 Tabulka Ceny

Tabulka Ceny byla vytvořena k účelu nadefinování ceníku pro jednotlivé sekce. To tedy znamená, že jsou tabulky mezi sebou provázané na základě indexů, konkrétně sloupec Id z tabulky Sekce odpovídá sloupci Id_sekce z tabulky Ceny (Obrázek 14). Každý řádek představuje jednu variantu z ceníku, například přístup na dvě hodiny s cenou 50 korun. Tabulka má dále definovány tyto sloupce:

- PocetJednotek určuje rozsah přístupu do dané sekce
- Cena ukládá informaci o ceně varianty
- DatovyTyp slouží k upřesnění, v jakém formátu se sloupec PocetJednotek nachází, může to být „Time“ pro časové omezení (v hodinách), „Int“ určující počet vstupů, nebo „Infinity“ k neomezenému vstupu

	Column Name	Data Type	Allow Nulls
🔑	Id	int	☐
	Id_sekce	int	☐
	PocetJednotek	float	☑
	Cena	float	☐
	DatovyTyp	nvarchar(10)	☐

Obrázek 14 Databázový slovní tabulky Ceny

4.3.3 Tabulka Vstup

Proces využívající párování obličejů při vstupu a výstupu se opírá o tabulku Vstup (Obrázek 15). Obsahuje stejných sedm členů pro určení relativních vzdáleností částí obličeje jako třída cOblicej. Hodnota těchto sloupců je vložena při vstupu uživatele a při výstupu se hodnoty pouze porovnávají, nikam se již neukládají. Kromě toho tabulka ještě obsahuje sloupce CasVstupu a CasVystupu pro časové určení vstupu a výstupu, Vystup usnadňující proces spárování, CenaCelkem uchovávající cenu, kterou uživatel zaplatil, a sloupec Foto pro uchování fotky při vstupu a možnosti řešení případného problému.

	Column Name	Data Type	Allow Nulls
🔑	Id	int	☐
	Foto	image	☐
	SirkaVyska	float	☐
	NosVyska	float	☐
	CeloVyska	float	☐
	BradaVyska	float	☐
	NosSirka	float	☐
	PusaSirka	float	☐
	BradaSirka	float	☐
	CasVstupu	datetime	☐
	CasVystupu	datetime	☑
	Vystup	bit	☐
	CenaCelkem	float	☐

Obrázek 15 Databázový slovník tabulky Vstup

4.3.4 Tabulka VybraneSekce

Výběry sekcí provedené uživatelem obsahuje tabulka VybraneSekce. Jejím cílem je propojení identifikátorů z tabulek Sekce, Ceny a Vstup (Obrázek 16). Je nutné si uvědomit, že při výběru určitého počtu sekcí bude tato tabulka obsahovat stejný počet

řádků týkajících se vždy stejného vstupu, avšak každý řádek se bude týkat jiné sekce a její vybrané varianty.

	Column Name	Data Type	Allow Nulls
🔑	Id	int	☐
	Id_vstup	int	☐
	Id_sekce	int	☐
	Id_ceny	int	☐

Obrázek 16 Databázový slovník tabulky VybraneSekce

4.4 Práce s Kinectem

Důvodů využití oficiálních knihoven je několik. Instalace je stejně jednoduchá jako nainstalování kteréhokoliv jiného programu, její součástí jsou i ovladače na samotný Kinect a také Kinect for Windows Developer Toolkit, který obsahuje celou řadu ukázek pro demonstraci využití zařízení a také jejich zdrojové kódy. Další výhodou je, že sám Microsoft a další nakladatelství vydaly několik publikací, které se této problematice věnují, a díky velkému rozšíření tohoto SDK lze najít v internetových diskuzích řešení mnoha problémů.

4.4.1 Připojení Kinectu k počítači

Pro přístup k datům z Kinectu je nejprve nutné se k němu připojit. To je možné hned dvěma způsoby. Třída KinectSensor obsahuje kolekci senzorů, které je nutné projít a otestovat stav připojení. V případě, že byl nalezen připojený senzor, můžeme s ním začít pracovat.

```
KinectSensor mujSensor;
foreach (var kinect in KinectSensor.KinectSensors)
{
	if (kinect.Status == KinectStatus.Connected)
	{
		mujSensor = kinect;
	}
}
```

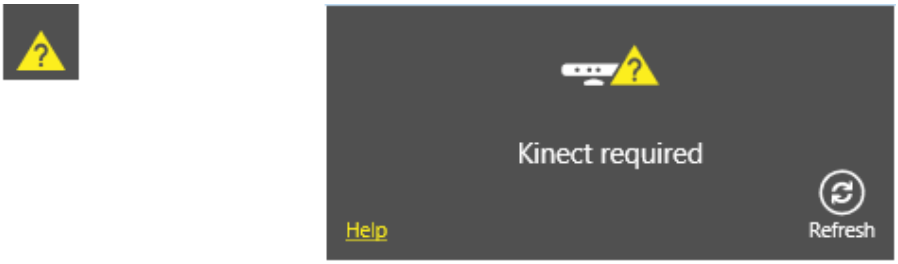
Ukázka kódu 2 Připojení senzoru pomocí průchodu kolekce

Druhý způsob je poněkud elegantnější. Lze totiž využít třídu SensorChooser, která sama vyhledává připojený senzor. Toto vyhledávání lze zapnout pomocí metody Start nebo zastavit metodou Stop. Také je možno vytvořit událost, která se provede při změně připojení, je tedy snadné ukončit práci se starým senzorem a ihned po připojení začít pracovat s novým.

```
KinectSensorChooser sensorChooser = new KinectSensorChooser();
sensorChooser.KinectChanged += sensorChooser_KinectChanged;
sensorChooser.Start();
```

Ukázka kódu 3 Využití SensorChooseru pro připojení k senzoru Kinect

Microsoft šel ve svém SDK ještě dále a nabízí k použití SensorChooserUI. Jak název napovídá, jde o komponentu, která je součástí uživatelského rozhraní, a je tedy viditelná i pro uživatele (Obrázek 17). Sama obsahuje SensorChooser, takže plní úkol naprosto stejně, ale navíc informuje uživatele o stavu připojení. Pokud k počítači není připojen žádný Kinect, nebo se vyskytne jiná chyba, upozorňuje na problém v podobě malého čtverce a při najetí na komponentu kurzorem se zvětší a poskytne více informací. V případě správného připojení zcela zmizí a nezabírá zbytečně místo.



Obrázek 17 SensorChooserUI - upozornění na chybu připojení

4.4.2 Streamy

Přístup k surovým datům ze zařízení je možný pomocí takzvaného streamu, tedy neustálým proudem dat od zdroje. K dispozici máme tři streamy – ColorStream obsahující data z RGB kamery, DepthStream zpřístupňující data z hloubkového senzoru a SkeletonStream, který je výsledkem algoritmu rozpoznávání postav před senzorem. Se všemi třemi streamy lze pracovat samostatně, nebo využít události, která je vyvolána v momentě, kdy jsou data ze všech tří snímků k dispozici. Záleží tedy na vývojáři, zdali potřebuje všechna data, nebo pouze některá z nich. Tato událost ale představuje velkou zátěž na výpočetní výkon, a proto není vhodné ji používat v momentech, kdy všechna data nejsou využívána.

ColorStream

Přístup ke ColorStreamu je možný až po vytvoření události ColorFrameReady a po jeho zapnutí. Zapnutí streamu se provede zavoláním metody Enable, která při neuvedení parametru nastaví rozlišení na 640 pixelů × 480 pixelů při 30 snímcích za sekundu. Zde musí vývojář volit mezi lepší kvalitou pořízeného snímku a lepší odezvou. Pokud se

rozhodne pro jinou variantu, předá metodě Enable parametr datového typu ColorImageFormat, který je výčtovým typem a obsahuje seznam možných nastavení.

Po zhotovení snímku kamerou dojde k vyvolání nastavené události a pomocí metody OpenColorImageFrame instance třídy ColorImageFrameReadyEventArgs lze ke ColorStreamu přistoupit a snímek získat.

```
void kinectSensor_ColorFrameReady(object sender,
ColorImageFrameReadyEventArgs e)
        {
            using (ColorImageFrame snimek = e.OpenColorImageFrame())
            {
                //prace se snimkem
            }
        }
```

Ukázka kódu 4 Přístup ke snímku z RGB kamery

DepthStream

DepthStream se ovládá velice podobně jako ColorStream, zapnutí streamu probíhá obdobně, opět je možné volit mezi kombinacemi rozlišení a počtu snímků za sekundu, přístup ke snímku je možné prostřednictvím metody OpenDepthImageFrame instance třídy DepthImageFrameReadyEventArgs. Stream navíc ale obsahuje jedno specifické nastavení. V závislosti na předpokládaném využití a prostředí lze stream nastavit na takzvaný Near mód, který pracuje ve vzdálenosti 0,4 metru–3 metry od senzoru, nebo na Default mód, čímž svůj rozsah posune do vzdálenosti 0,8 metru–3,5 metru. Mimo tyto vzdálenosti není Kinect schopný správně pracovat.

```
kinectSensor.DepthStream.Range = DepthRange.Near;
kinectSensor.DepthStream.Range = DepthRange.Default;
```

Ukázka kódu 5 Nastavení vzdálenosti, ve které bude hloubkový senzor pracovat

SkeletonStream

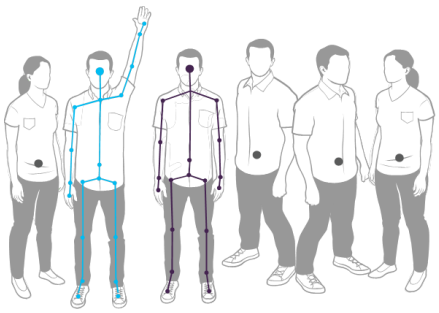
Tento stream se od ostatních dvou výrazně liší, protože data od něj nevycházejí přímo ze senzoru, ale jsou výsledkem zpracování dat z hloubkového senzoru a částečně i z RGB kamery. Výsledkem tohoto procesu jsou data s informacemi o jednotlivých uživatelích a jejich částech těla. Zapnutí streamu se opět provádí metodou Enable, která ale v případě uvedení parametru očekává datový typ TransformSmoothParameters. Tento parametr je využíván k hladším změnám z minulého snímku, výsledný pohyb je tedy plynulejší a odfiltrují se případné krátkodobé chyby.

Pro získání dat o uživatelích je potřeba v události získat data ze snímku a zkopírovat je pomocí metody CopySkeletonDataTo instance třídy SkeletonFrame do datového pole typu Skeleton.

```
using (SkeletonFrame skeletonSnimek = e.OpenSkeletonFrame())
{
	if (skeletonSnimek != null && this.skeletonPole != null)
	{
		skeletonSnimek.CopySkeletonDataTo(this.skeletonPole);
	}
}
```

Ukázka kódu 6 Zkopírování dat ze snímku do datového pole typu Skeleton

Po zkopírování dat získáme pole o šesti prvcích, kde každý prvek reprezentuje jednoho uživatele. V případě, že se před senzorem nepohybuje tolik lidí, mají zbylé prvky hodnotu null, tedy prázdnou hodnotu. Z omezení, která ze senzoru vychází, je možné pouze sledovat dvě osoby detailně, u zbylých lze určit pouze pozici v zorném poli (Obrázek 18).



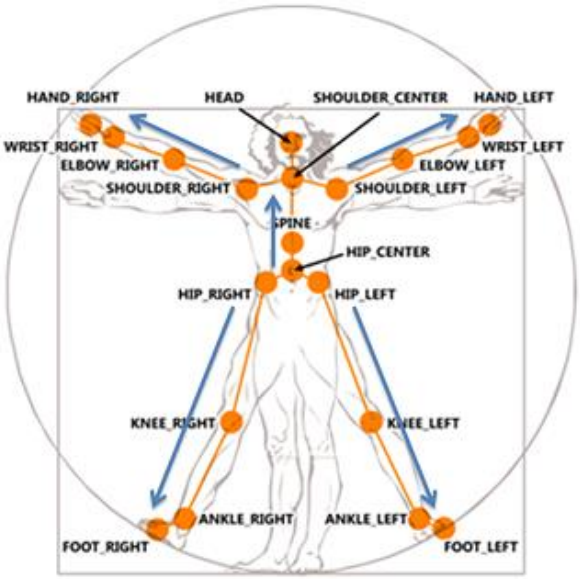
Obrázek 18 Rozpoznání postav Kinectem („Skeletal Tracking", 2012)

Aby bylo možné poznat, který z prvků pole obsahuje detailní informace o částech těla, je možné otestovat hodnotu vlastnosti TrackingState. Ta může nabývat tří hodnot, NotTracked když nedojde k vysledování postavy, PositionOnly v případě, že byla určena pouze poloha uživatele a Tracked, pokud došlo k vysledování postavy i jejích částí.

```
foreach (Skeleton skeleton in skeletonPole)
{
	if (skeleton.TrackingState == SkeletonTrackingState.Tracked)
	{
		//práce s postavou i jejími částmi
	}

	if (skeleton.TrackingState == SkeletonTrackingState.PositionOnly)
	{
		//práce pouze s postavou
	}
}
```

Ukázka kódu 7 Testování pole možných uživatelů

Pokud došlo k rozpoznání částí postavy, je možné využít kolekci Joints, která obsahuje jednotlivé body lidského těla. Počet těchto bodů záleží na nastavení sledovacího módu. Při nastavení vlastnosti SkeletonTrackingMode na hodnotu Seated je k dispozici celkem deset bodů. Tento mód totiž předpokládá, že uživatel sedí, a proto by bylo velice obtížné určit pozice spodních částí těla. Oproti tomu Default mód počítá s tím, že uživatel před senzorem stojí, je dále od pozadí, a tak je filtrace postavy od okolí snadnější. Při optimálních podmínkách dokáže určit dvacet částí postavy (Obrázek 19), tedy stejných deset jako při Seated módu a dalších deset, které se vztahují ke spodní části těla.



Obrázek 19 Části lidského těla, které Kinect rozlišuje („Tracking Users with Kinect Skeletal Tracking", 2012)

To ovšem neznamená, že vždy dochází k určení všech bodů. Může se totiž stát, že uživatel je částí těla mimo záběr, a proto není možné určit jeho polohu. Kvůli tomu obsahuje každý Joint vlastnost TrackingState, jejíž hodnota určuje, jestli byl daný bod vysledován či nikoliv. Pokud se rovná hodnotě NotTracked, znamená to, že poloha Jointu nebyla vůbec zjištěna. V případě, že je roven hodnotě Inferred, došlo k odvození polohy bodu. To se může stát například, když má uživatel předpaženou ruku a dochází k zákrytu ramene. Kinect není schopný přesně určit polohu, ale má hrubou představu o jeho pozici a ta je programátorovi i přesto k dispozici. Nejideálnější je situace, kdy se vlastnost rovná hodnotě Tracked a to znamená, že určení tohoto bodu bylo úspěšné.

```
if (skeleton.Joints[JointType.HandRight].TrackingState ==
JointTrackingState.Tracked)
{
      //práce s pravou rukou, která byla vysledována
}

if (skeleton.Joints[JointType.HandRight].TrackingState ==
JointTrackingState.Inferred)
{
      //práce s pravou rukou, která byla odvozena
}
```

Ukázka kódu 8 Testování stavu pravé ruky

K informaci o pozici slouží vlastnost Position, která je datového typu SkeletonPoint, a ten obsahuje souřadnice os X, Y a Z, přičemž X je vodorovná osa, Y svislá osa a Z je prostorová osa.

Pokud má programátor zájem sledovat jednotlivé body na obličeji, musí přidat do svého projektu referenci na FaceTracking z SDK. Tímto se mu zpřístupní knihovny, které mu pomohou dosáhnout výsledku stejně snadno, jako to bylo u získávání dat o uživatelích. Postup začíná naprosto stejně, jenom je potřeba získat data ze všech streamů. Proto se doporučuje využití události, která se provádí v okamžiku, kdy jsou všechny snímky hotové. Ta totiž zaručuje, že snímky jsou pořízené ve stejném okamžiku, při použití jednotlivých událostí pro každý stream není zaručeno, že se data budou týkat stejné situace. Nejprve je nutné získat ze streamů dané snímky a zkopírovat si je do příslušných polí.

```
//ziskani dat ze streamu
ColorImageFrame colorSnimek = e.OpenColorImageFrame();
DepthImageFrame depthSnimek = e.OpenDepthImageFrame();
SkeletonFrame skeletonSnimek = e.OpenSkeletonFrame();
//deklarace a inicializace poli ze snimku
byte[] colorObrazek = new byte[colorSnimek.PixelDataLength];
short[] depthObrazek = new short[depthSnimek.PixelDataLength];
Skeleton[] skeletonData = new Skeleton[skeletonSnimek.SkeletonArrayLength];
//zkopirovani dat do poli
colorSnimek.CopyPixelDataTo(colorObrazek);
depthSnimek.CopyPixelDataTo(depthObrazek);
skeletonSnimek.CopySkeletonDataTo(skeletonData);
```

Ukázka kódu 9 Získání a zkopírování dat ze všech snímků

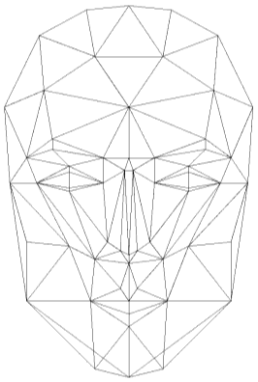
Dále se musí deklarovat proměnná typu FaceTracker a vytvořit její instance, konstruktor požaduje jako parametr instanci aktuálně snímaného Kinectu. Při procházení pole datového typu Skeleton, stejně jako tomu bylo u sledování postavy, stačí zavolat metodu

Track instance třídy FaceTracker se všemi potřebnými parametry, tedy daty a formáty všech snímků, a její návratovou hodnotu přiřadit do proměnné datového typu FaceTrackFrame. Hodnota její vlastnosti TrackSuccessful totiž obsahuje informaci, která udává, jestli bylo vysledování obličeje úspěšné. Pro získání finálních dat stačí zavolat metodu GetProjected3DShape instance FaceTrackFrame, která vrací kolekci bodů obličeje.

```
FaceTracker faceTracker = new FaceTracker(sensor);
foreach (Skeleton skeleton in skeletonData)
{
       //zpracovani vsech snimku a ziskani dat o obliceji
       FaceTrackFrame faceTrackerFrame =
       faceTracker.Track(colorSnimek.Format, colorObrazek,
       depthSnimek.Format, depthObrazek, skeleton);

       //pokud se prevod dat povedl
       if (faceTrackerFrame.TrackSuccessful)
       {
       this.oblicej = new cOblicej(faceTrackerFrame.GetProjected3DShape(),
       colorObrazek);

       }

}
```

Ukázka kódu 10 Facetracking, získání bodů obličeje

K dispozici je celkem 121 bodů, které vycházejí z obličejového modelu Candide (Obrázek 20). Většina z nich je slovně upřesněna v definici výčtového typu FeaturePoint, některé jsou ale nepopsané.



Obrázek 20 Parametrizovaný model obličeje Candide

4.4.3 Ovládání kurzoru rukou

Od verze SDK 1.7 může vývojář využít ovládací prvek KinectRegion, který mu nesmírně ulehčuje práci, protože tento prvek je stačí pouze umístit, nadefinovat jeho velikost a

přiřadit vlastnosti KinectSensor aktivní sensor. Pokud Kinect rozpozná člověka s předpaženou rukou a nataženou dlaní, začne pohybovat kurzorem v závislosti na jeho dlani. Aby bylo více zdůrazněno ovládání rukou, změní se klasický kurzor na větší obrys natažené dlani. Pro ovládání je možné použít levou i pravou ruku bez nutnosti nějakého nastavování.

4.4.4 Zobrazení uživatele

Bez zpětné vazby Kinectu je občas těžké určit, proč nereaguje na pohyby a uživateli tak nezbývá, než hádat, v čem je problém. To dokáže vyřešit komponenta KinectUserViewer, která zobrazuje data z hloubkového senzoru a odfiltrováním pozadí vykresluje siluetu postavy. Uživatel tak vidí, zda stojí v zorném poli, nebo má svoji pozici přizpůsobit. Použití se často kombinuje s již zmíněným KinectRegionem, protože při detekci ruky, která má KinectRegion ovládat, silueta změní barvu a uživatel má opět užitečnou informaci o aktuálním stavu dění (Obrázek 21). Všechny barvy je možné nastavit pomocí odpovídajících vlastností.

Obrázek 21 KinectUserViewer zobrazující siluetu postavy uživatele

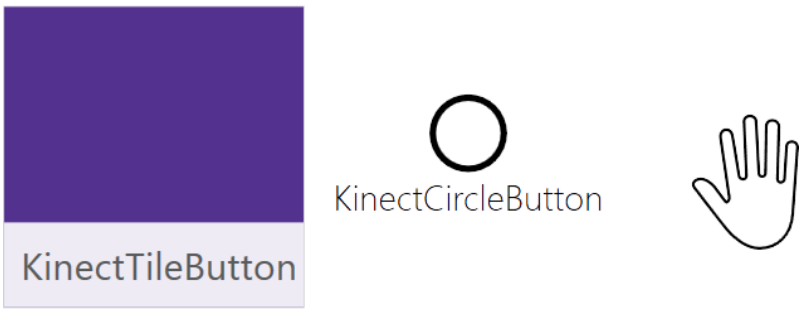
4.4.5 Ovládací prvky

Protože běžné ovládací prvky z klasického uživatelského rozhraní nejsou vhodné pro použití v přirozeném uživatelském rozhraní, obsahuje SDK i takové komponenty, které jsou tomu lépe uzpůsobeny (Obrázek 22). K dispozici jsou dvě tlačítka (Obrázek 23) a jeden posuvník (Obrázek 24).

Obrázek 22 Srovnání klasických ovládacích prvků a kurzoru

KinectTileButton a KinectCircleButton

Tato dvě tlačítka se od sebe příliš neliší. Největší rozdíl mezi nimi je ve vzhledu. Už podle názvu lze vyvodit, že KinectTileButton má tvar dlaždice a kromě popisku může obsahovat i obrázek na pozadí. V případě, že obrázek není nastaven, vykreslí se pozadí nastavenou barvou. Oproti tomu KinectCircleButon má tvar kruhu a popisek se nachází pod ním. Do zmíněného kruhu lze umístit ještě další text. Oběma těmto tlačítkům lze nastavit událost Click, která se provede při stisku tlačítka. Uživatel tedy na něj musí kurzorem najet, pohybem dlaní od sebe tlačítko virtuálně stisknout a pohybem k sobě tlačítko uvolnit. V případě, že tlačítko neuvolní, KinectRegion toto gesto vyhodnotí jako zrušení stisku a daná akce se neprovede.



Obrázek 23 Tlačítka KinectTileButton a KinectCircleButton

KinectScrollViewer

Tato komponenta je užitečná v případě, že zobrazovací plocha není dostatečně velká a je potřeba využít posuvníku. Při ovládání myší si uživatelé zvykli otáčet kolečkem u myši, při ovládání kurzoru rukou musí uživatel najet do prostoru posuvníku, který změní barvu a dá tak uživateli najevo, že obsahuje i komponenty, které nejsou aktuálně viditelné. Pro posunutí sevře uživatel dlaň v pěst, tím virtuálně chytí pomyslný posuvník, a pohybem ruky posouvá obsah.

Obrázek 24 Uchopení posuvníku sevřením dlaně v pěst

Směr uspořádání prvků nezávisí přímo na KinectScrollVieweru. Tomu se totiž do vlastnosti Content přiřazuje jiná komponenta, která má toto na starosti. Nejvíce se pro to

hodí WrapPanel. Ten umožňuje nastavení vertikálního nebo horizontálního řazení a pro přidání ovládacího prvku je potřeba zavolat metodu Add vlastnosti Children, které se jako parametr předá potomek pro vložení.

```
wrapPanel.Children.Add(new KinectCircleButton());
kinectScrollViewer.Content = wrapPanel;
```

Ukázka kódu 11 Vložení tlačítka do komponenty KinectScrollViewer

4.5 Popis User Controlů a hlavního okna

4.5.1 Hlavní okno (MainWindow)

Hlavní okno zajišťuje koordinaci a spolupráci mezi jednotlivými prvky systému. Při inicializaci komponenty v konstruktoru MainWindow dochází také k vytvoření instancí všech user controlů, přidání těchto instancí do Gridu gridKinectRegion pro pozdější zobrazování, a třídy cDatabaze, které je předán takzvaný Connection String pro přístup k databázovému serveru (více popsáno v kapitole 4.2.1), a vytvoření události při načtení okna MainWindow_Loaded.

Okno je rozděleno do dvou sloupců. V první části je umístěn KinectSensorChooserUI pro informování uživatele o stavu připojení a KinectUserViewer pro zobrazení uživatele z pohledu senzoru. Tyto dva prvky se překrývají, ale reálně nemůže nastat situace, kdy by byly zobrazeny oba. V případě správného připojení senzoru Kinect k počítači se KinectSensorChooserUI zneviditelní a KinectUserViewer má tak prostor pro zobrazení svých dat. Pokud dojde k chybě v připojení, nemá KinectUserViewer k dispozici žádná data a nepřekáží tak komponentě KinectSensorChooserUI. Posledním prvkem je tlačítko umístěné v pravém horním rohu pro zobrazení user controlu s nastavením. Ve druhé části okna je umístěna komponenta KinectRegion, která zajišťuje ovládání aplikace pomocí uživatelovy ruky (popsáno v kapitole 4.4.3). Jeho vlastnost Content obsahuje Grid gridKinectRegion, který slouží k zobrazování jednotlivých user controlů.

Přepínání viditelnosti zpracovává metoda ZmenitUserControl, která požaduje dva parametry, první výčtového typu MojeUserControly pro určení, který z user controlů má být nyní zobrazen, a druhý datového typu UserControl pro schování stávajícího user controlu. Pokud druhý parametr obsahuje hodnotu (není roven null), pak je schování provedeno nastavením jeho vlastnosti Visibility na hodnotu Visibility.Hidden. V případě, že parametr obsahuje hodnotu null, je nutné user controly schovat všechny. Výčtový typ

MojeUserControly obsahuje seznam pohledů, které je možné zobrazit. Na základě této hodnoty se nastaví příslušnému user controlu vlastnost Visibility na hodnotu Visibility.Visible a v případě, že má implementovanou metodu AktivaceControlu pro přípravu dat před zobrazením, dojde k jejímu zavolání ještě před zviditelněním.

```
foreach (var child in gridKinectRegion.Children)
        if (child is UserControl)
                ((UserControl)child).Visibility = System.Windows.Visibility.Hidden;
```

Ukázka kódu 12 Zneviditelnění všech user controlů

Zásadní pro přístup k datům ze senzoru Kinect je proměnná sensorChooser datového typu KinectSensorChooser (detailněji popsán v kapitole 4.4.1), jejíž instance je vytvořena v události MainWindow_Loaded, a od tohoto okamžiku začíná hledat připojený Kinect. V okamžiku změny připojení, tedy připojení i odpojení, je vyvolána událost sensorChooser_KinectChanged, která se stará o vypnutí nebo zapnutí všech streamů a nastavení instance KinectSensor komponentě KinectRegion pro aktivaci ovládání rukou. Protože přístup k surovým datům ze senzoru je nutné pouze pro rozpoznávání obličeje, předávají se nové snímky user controlu uOblicej pouze v případě, že je viditelný. Zastavení hledání nového senzoru je provedeno v události při zavření okna Window_Closed.

Událost NewSensor_AllFramesReady, která je vyvolána při vytvoření všech snímků ze senzoru, je vytvářena nebo rušena voláním metod VytvorUdalostSnimky a ZastavUdalostSnimky.

4.5.2 User Control uVstupVystup

Po spuštění aplikace dojde k zobrazení tohoto user controlu určujícího, zda uživatel požaduje vstup či výstup. K tomu slouží dvě tlačítka KinectTileButton. Protože se zde nenacházejí žádné další prvky, je prostor pro zobrazení rozdělen pomocí Gridu na dva sloupce s šířkou „1*“ (tím dojde k rozdělení Gridu na dva stejně velké sloupce) a tlačítka jsou umístěna zarovnána na střed v horizontálním i vertikálním směru. Obě mají velikost 400 pixelů × 400 pixelů pro snadné ovládání. Při stisku tlačítka btnVstup určeného pro vstup dojde k zobrazení user controlu uSekce (více v následující kapitole 4.5.3). Stisknutím tlačítka btnVystup se předají user controlu uOblicej (popsán v kapitole 4.5.5) informace, za jakým účelem je zobrazován (v tomto případě výstupu), a dojde k jeho zviditelnění.

```
(Application.Current.MainWindow as
MainWindow).ZmenitUserControl(MojeUserControly.Sekce, this);
```

Ukázka kódu 13 Zobrazení user controlu uSekce

4.5.3 User Control uSekce

Grid rozděluje tento user control na dva řádky. Horní má nastavenou automatickou výšku podle svého obsahu a je v něm umístěn Label zobrazující celkovou cenu, kterou má uživatel zaplatit. Spodní část je určena pro tlačítka typu KinectTileButton reprezentující jednotlivé sekce. Protože se může stát, že prostor pro zobrazení nemusí být vždy dostatečný, jsou tlačítka umístěna pomocí WrapPanelu v komponentě KinectScrollViewer, aby byla zajištěna možnost posouvání.

Metoda AktivaceControlu připraví user control na zobrazení tím, že načte dostupné sekce z databáze spolu s jejich variantami ceníku, vytvoří z nich instance třídy cSekce, vygeneruje k nim příslušná tlačítka KinectTileButton (pojmenuje je podle názvu sekce, přiřadí jim obrázek na pozadí, vytvoření jim události po stiknutí button_Click a podobně) a instance třídy cSekce je přiřazena do vlastnosti Tag příslušného tlačítka. Všechna tlačítka jsou přidána do komponenty WrapPanel pro zviditelnění a jako poslední dojde k vytvoření tlačítka pro potvrzení výběru. To se od ostatních liší tím, že je mu vytvořena událost po stisknutí buttonPotvrdit_Click.

```
for (int i = 0; i < tabulka.Rows.Count; i++)
{
       //vygenerovani tlacitka
       button = new KinectTileButton();
       button.Tag = sekce;
       button.Label = sekce.Nazev;
       button.LabelBackground = Brushes.WhiteSmoke;
       button.Click += button_Click;;

       //pridani tlacitka do wrapu
       this.wrapSekce.Children.Add(button);
}
```

Ukázka kódu 14 Vygenerování tlačítek reprezentující sekce

K rozpoznání vybrané sekce dochází na základě určení barvy pozadí popisku tlačítek, které sekce reprezentují. Pokud se hodnota rovná WhiteSmoke, znamená to, že uživatel chce sekci vybrat, a proto dojde k předání tlačítka user controlu uCenik a přepnutí pohledu na něj. Když je hodnota pozadí jiná, sekce byla v minulosti vybrána a uživatel ji chce nyní odstranit ze svého výběru. Barva pozadí se nastaví zpět na WhiteSmoke, instanci třídy cSekce v tagu tlačítka je vlastnosti VybranaVariantaId na prázdnou hodnotu (null)

a zavolá se metoda UkazCelkovouCenu změnu labelu s celkovou cenou. Tato metoda využívá vlastnosti CenaCelkem, která prochází všechny potomky ve WrapPanelu a přepočítává celkovou cenu.

```
double cena = 0;
foreach (var child in this.wrapSekce.Children)
{
	if (child is KinectTileButton && (child as KinectTileButton).Tag != 
	null && ((child as KinectTileButton).Tag as cSekce).VybranaVarianta
	!= null)
	{
		cena += ((child as KinectTileButton).Tag as
		cSekce).VybranaVarianta.Cena;
	}
}
```

Ukázka kódu 15 Výpočet celkové ceny

Pro potvrzení nebo zrušení výběru sekce je volána metoda PotvrditVolbu, která vyžaduje dva parametry, první je typu KinectTileButton a druhý je logická hodnota. Druhý parametr udává, jestli byl výběr varianty z ceníku úspěšný, pokud ano, dojde ke změně barvy pozadí tlačítka a nakonec se zmiňovaná metoda UkazCelkovouCenu.

4.5.4 User Control uCenik

Pro zobrazení ceníku konkrétní sekce slouží tento user control. Je opět rozdělen na dva řádky, v prvním se zobrazují tlačítka KinectTileButton, reprezentující varianty ceníku, a ve druhém řádku je umístěné tlačítko KinectTileButton, vracející pohled zpátky na user control uSekce bez výběru žádné z variant. Horní část obsahuje KinectScrollViewer, pro opětovné zajištění možnosti viditelnosti všech tlačítek.

Přípravu na zobrazení provádí metoda NaplnCenik, která má jeden parametr typu KinectTileButton představující tlačítko sekce, které bylo stisknuto v user controlu uSekce. Tlačítko je přirazeno do globální proměnné zmacknuteTlacitko a následně dojde k naplnění komponenty WrapPanel vygenerovanými tlačítky s variantami ceníku. Kliknutím na jedno z těchto tlačítek dojde k nastavení vybrané varianty do tagu proměnné zmacknuteTlacitko, zavolání metody PotvrditVolbu z user controlu uSekce a přepnutí pohledu na něj.

4.5.5 User Control uOblicej

User control uOblicej zpracovává rozpoznání obličeje. Rozhraní je rozděleno na dva sloupce, v levém sloupci je tlačítko KinectTileButton, pro přepnutí pohledu na user

control uVstupVystup, v pravém je umístěna komponenta Image, která zobrazuje výstup z RGB kamery, a textové pole, které je primárně schováno a po úspěšném rozpoznání obličeje se zobrazí s krátkou zprávou.

Implementovaná metoda AktivaceControlu zajišťuje nastavení proměnných na počáteční hodnoty před samotným zobrazením user controlu. Nastaví tedy proměnnou oblicej typu cOblicej na prázdnou hodnotu (null), zavolá metodu VytvorUdalostSnimky instance MainWindow pro vytvoření události při pořízení snímků senzorem a schová textové pole se zprávou.

K rozpoznání obličeje dochází v metodě Sensor_AllFramesReady, která má čtyři parametry – snímek z ColorStream, snímek z DepthStream, snímek z SkeletonStrem a senzor Kinect. Po kontrole neprázdnosti jednotlivých streamů dojde k zobrazení aktuálního snímku z ColorStream v komponentě Image (uživatel se tak vidí na obrázku) a začne proces rozpoznání obličeje popsaný v kapitole 4.4.2 a podkapitole SkeletonStream. Pokud je user control v režimu vstup, vloží se informace o obličeji vybraných sekcí uživatelem do databáze a zobrazí se textové pole s krátkou zprávou. V případě výstupu je nutné získaná data porovnat s obličeji, které jsou již v databázi. Srovnávání se ale týká pouze těch řádků, které mají hodnotu Vystup nastavenou na false, aby nedocházelo ke zbytečnému porovnávání se vstupy, u nichž byl výstup potvrzen. Párování začíná s odchylkou 0,05, která se při neúspěchu nalezení shody zvyšuje o stejnou hodnotu. Pokud dojde ke shodě s jedním záznamem z databáze, spárování proběhlo úspěšně. Může se ale stát, že nenastane žádná shoda a je potřeba, aby se se rozpoznání opakovalo, nebo může být nalezena shoda s více záznamy a k takovému řešení už je potřeba lidského zásahu. Po úspěšném rozpoznání obličeje se událost vyvolávající při pořízení snímků zruší zavoláním metody ZastavUdalostSnimky instance MainWindow.

Aby si stihl uživatel textovou zprávu přečíst, je v momentě jejího zobrazení spuštěn odpočet pěti sekund a po jeho uplynutí dojde k přepnutí pohledu zpět na user control uVstupVystup.

4.5.6 User Control uNastaveni

User control je určen pouze pro ovládání myší, protože obsahuje klasické ovládací prvky. Jeho smyslem je správa sekcí, které jsou aktuálně v databázi. Načtení dat se provede zavoláním metody AktivaceControlu, která připraví user control pro zobrazení. Přístup

k datům je zajištěn zavoláním metody DotazRadky instance třídy cDatabaze, která je vytvořena v hlavním okně programu. Získané řádky z databáze použije jako parametr pro konstruktory při vytváření listu s prvky typu cSekce. Tento list je zdrojem prvků pro ListView, které zajistí jejich zobrazení.

Při dvojkliku na některý z prvků v ListView je tento prvek předán user controlu uSekceUprava a přepnutí pohledu na něj pro editaci sekce. Kliknutím na tlačítko btnPridatSekci dojde k zobrazení toho samého user controlu, ale v režimu pro přidávání.

Tlačítko btnZpet zajišťuje přepnutí pohledu na uVstupVystup.

4.5.7 User Control uSekceUprava

Tento jednoduchý user control slouží pro přidávání, upravování a mazání sekcí z databáze. Tomu odpovídá i grafické rozhraní, které obsahuje textová pole a komponentu Image kopírující strukturu třídy cSekce.

Při zobrazování je volána metoda AktivaceControlu, která v módu vytváření vymaže všechna textová pole a znemožní stisknutí tlačítka pro btnSmazat. Pro úpravu sekce je volána metoda NastavSekciProUpravu, které je předán parametr typu cSekce, a provede naplnění polí příslušnými hodnotami. Po kliknutí na tlačítko btnPotvrdit potvrzující změnu či vytvoření se nejprve provede kontrola zadaných údajů metodou Kontrola. Pokud je vše v pořádku, dojde k vytvoření SQL dotazu, který je následně předán metodě UpdateRadky nebo VlozitRadky instance třídy cDatabaze, a ten se následně provede.

Při vytváření dotazu pomáhají metody HodnotaNeboNULL, která nahrají prázdný řetězec ("") prázdnou hodnotou (null) a ObrazekDoBytePole, která převádí obrázek do bytového pole, aby bylo možné ho vložit do databáze.

5 Závěr

Vhodnost použití senzoru Kinect pro terminál a identifikování návštěvníků na základě rozpoznávání obličeje je závislá na podmínkách, ve kterých by měl pracovat. V případě ideálních podmínek, kdy je eliminováno přímé sluneční záření, zajištěno dobré umělé osvětlení obličeje a uživatelé se při rozpoznání obrazu dívají stejným směrem, je přesnost relativních vzdáleností poměrně velká. To ostatně ukazuje Příloha B, která znázorňuje tři různé pokusy rozpoznání obličeje v téměř stejné poloze.

Tyto podmínky je ale v praxi těžké zajistit, takže dochází k většímu zkreslení dat, než by bylo žádoucí. V takovém případě není možné získaná data ihned porovnávat a hledat shodu, ale využít některé z metod, které se běžně k rozpoznávání obličeje používají. To se nabízí jako ideální kombinace, protože při získání takového množství bodů obličeje, které je senzor Kinect schopen určit, a aplikování některé z metod face recognizingu by se úspěšnost mohla zvýšit. Tato úspěšnost ovšem nebude nijak závratně vysoká, protože nejlepší současné metody se pohybují okolo hodnoty 85 % (Hsu, 2013).

Také je potřeba brát v úvahu návštěvnost, která fungování terminálu dokáže ovlivnit. Při větší aktuální návštěvnosti se zvětšuje pravděpodobnost, že někteří návštěvníci budou mít velmi podobné rysy a o to větší budou kladeny nároky na jejich odlišení.

V tomto případě je zřejmě zapotřebí obsluhy, která by dokázala řešit případné problémy, a proto by bylo nutné model vylepšit a případně podpořit jiným bezpečnostním prvkem.

I Summary a keywords

The bachelor work's topic is utilization of input device XBOX Kinect. The theoretical part focuses on technical specialization, properties, single parts and history of this device and other motion sensors. It also describes the way of getting data and processing them, proposal of algorithms and flowcharts. The practical part examines developing application, which uses this motion sensor as a control device. That means that the program scans human body, evaluates the position of it, designates the gesture and does relevant action.

For coding, the application uses Visual Studio as an integrated developing environment, C# as a programming language, .NET Framework 4.5 as a software framework, Widnows Presentation Foundation and Windows Software Development Kit 2.0 as a communicator with the Kinect.

Key words: Kinect, motion sensors, human body tracking, face recognizing, motion controlling

II Seznam použitých zdrojů

Amazon.com: Wii: Nintendo Wii: Wii Hardware: Video Games. (2005). *Amazon.com*. Retrieved from: http://www.amazon.com/Wii-Nintendo/dp/B0009VXBAQ

Andrade, C., Livermore, S., Meyers, M., & Van Vliet, S. (2007). *Professional WPF programming: .NET development with the Windows Presentation Foundation.* (p. xxv, 451 p.). Indianapolis, IN: Wiley Pub..

Borenstein, G. (2012). *Making things see: 3D vision with kinect, processing, Arduino, and MakerBot.* (p. xviii, 416). Sebastopol,: O'Reilly.

Brain, M. (2007). How the Wii Works. *HOWSTUFFWORKS*. Retrieved from: http://electronics.howstuffworks.com/wii4.htm

Catuhe, D. (2012). *Programming with the Kinect for Windows Software Development Kit.* (p. xiv, 207 p.). Redmond, Wash.: Microsoft Press.

Desjardins, P. (2014). *Visual Studio Condensed.* New York: Apress.

Dewson, R. (2012). *Beginning SQL Server 2012 for developers.* (3rd ed., p. xx, 697 p.). New York: Distributed to the book trade worldwide by Springer Science Business Media.

Microsoft says Xbox 360 sales have surpassed 76 million units, Kinect sales top 24 million. (2013). *BGR*. Retrieved from: http://bgr.com/2013/02/12/microsoft-xbox-360-sales-2013-325481/

Fastest-selling gaming peripheral. (2011). *Guinness World Records*. Retrieved from: http://www.guinnessworldrecords.com/world-records/fastest-selling-gaming-peripheral/

Flatley, J. (2010). Visualized: Kinect + night vision = lots and lots and lots of dots (video). *Engadget*. Retrieved from: http://www.engadget.com/2010/11/08/visualized-kinect-night-vision-lots-and-lots-and-lots-of-do/

Garofalo, R. (2011). *Applied WPF 4 in context.* (p. xvi, 335 p.). New York: Distributed to the book trade worldwide by Speinger Science Business Media.

Giorio, C., & Fascinari, M. (2013). *Kinect in motion audio and visual tracking by example.* (New Edition.). S.l.: Packt Publishing Limited.

Hsu. (2013). How Face Recognition Tech Will Change Everything. *Discovery News*. Retrieved from: http://news.discovery.com/tech/biotechnology/how-face-rec-tech-change-everything-130611.htm

Jafri, R., & Arabnia, H. (2009). A Survey of Face Recognition Techniques. *Journal of Information Processing Systems*, *5*(2), pp. 41-68.

Jana, A. (2012). *Kinect for Windows SDK programing guide: build motion-sensing applications with Microsoft's Kinect for Windows SDK quickly and easily.* (New Edition.). Birmingham: [Packet] Publishing.

Johnson, B. (2014). *Professional visual studio 2013.* (1st edition., p. pages cm). Indiana: John Wiley & Sons, Inc..

Kališ, K. (2010). Test: Playstation Move – s Wii na ostří nože. *Doupě*. Retrieved from: http://doupe.zive.cz/clanek/test-playstation-move--s-wii-na-ostri-noze/?add=1

Kinect Launches a Surgical Revolution. (2012). *Microsoft Research*. Retrieved from: http://research.microsoft.com/en-us/news/features/touchlesssurgery-060712.aspx

Kongsro, J. (n. d.). Estimation of pig weight using a Microsoft Kinect prototype imaging system. pp. 32-35.

Košťál, F., & Sláma, D. (2010). Sci-fi do obýváku. *Computer: počítačový čtrnáctideník*, *17*(23), pp. 26-27.

Lee, J. (2008). Hacking the Nintendo Wii Remote. *IEEE Pervasive Computing*, *7*(3), pp. 39-45.

MacDonald, M. (2012). *Pro WPF 4.5 in C#: Windows Presentation Foundation with .NET 4.5.* (Fourth edition., p. xxxiii, 1078 pages). New York: Apress.

Malizia, A., & Bellucci, A. (2012). The artificiality of natural user interfaces. *Communications of the ACM*, *55*(3), pp. 36-.

Mistry, R., & Misner, S. (2014). *Introducing Microsoft SQL Server 2014.* (p. xv, 125 pages). Redmond, Wash: Microsoft Press.

Microsoft's Kinect guards South Korea's borders against trespassers. (2014). *Engadget*. Retrieved from: http://www.engadget.com/2014/02/04/microsoft-kinect-south-korea-dmz/

NPD: Total Industry Consumer Spending on Video Games at $4.6 Billion for Q1 2014. (2014). *NPD Group*. Retrieved from: https://www.npd.com/wps/portal/npd/us/news/press-releases/npd-total-industry-consumer-spending-on-video-games-at-4-billion-for-q1-2014/

Original Kinect for Windows sensor sales to end in 2015. (2014). *MSDN Blogs*. Retrieved from: http://blogs.msdn.com/b/kinectforwindows/archive/2014/12/30/original-kinect-for-windows-sensor-sales-to-end-in-2015.aspx

Parmar, D., & Mehta, B. (1970). Face Recognition Methods & Applications. *International Journal of Computer Technology and Applications*, *4*(1), pp. 84-86.

PlayStation®Move - PlayStation®3. (2014). *Playstation.com*. Retrieved from: http://br.playstation.com/ps3/playstation-move/

PLAYSTATION®MOVE MOTION CONTROLLER TO HIT WORLDWIDE MARKET STARTING THIS SEPTEMBER. (2010). *Sony Computer Entertainment Inc.*. Retrieved from: http://scei.co.jp/corporate/release/100616ae.html

PlayStation®Move Setup. (2013). *PlayStation Knowledge Center*. Retrieved from: https://support.us.playstation.com/app/answers/detail/a_id/2092/~/playstation%C2%AE move-setup

Rouse, M. (2011). Natural user interface (NUI). *WhatIs.com*. Retrieved from: http://whatis.techtarget.com/definition/natural-user-interface-NUI

Ruda, L. (2011). Kinect pentru PC?. *Zona.ro*. Retrieved from: http://zona.ro/2011/02/kinect-pentru-pc/

Sarkar, D. (2013). *Microsoft SQL Server 2012 with Hadoop.* Birmingham, UK: Packt Publishing.

Skeletal Tracking. (2012). *MSDN - the microsoft developer network*. Retrieved from: https://msdn.microsoft.com/en-us/library/hh973074.aspx

St. Jean, S. (2012). *Kinect hacks.* (1st ed., p. xiv, 262 p.). Sebastopol, CA: O'Reilly.

Thai, T., & Lam, H. (2002). *.NET framework essentials.* (2nd ed., p. 307 p.). Sebastol, CA: O'Reilly.

THE TECH BEHIND PLAYSTATION MOVE. (2010). *IGN*. Retrieved from: http://www.ign.com/articles/2010/09/18/the-tech-behind-playstation-move

Tracking Users with Kinect Skeletal Tracking. (2012). *MSDN - the microsoft developer network*. Retrieved from: https://msdn.microsoft.com/en-us/library/jj131025.aspx

Troelsen, A. (2012). *Pro C# 5.0 and the .NET 4.5 framework.* (Sixth edition., p. lxvii, 1487 pages). Berkeley, Calif.: Apress.

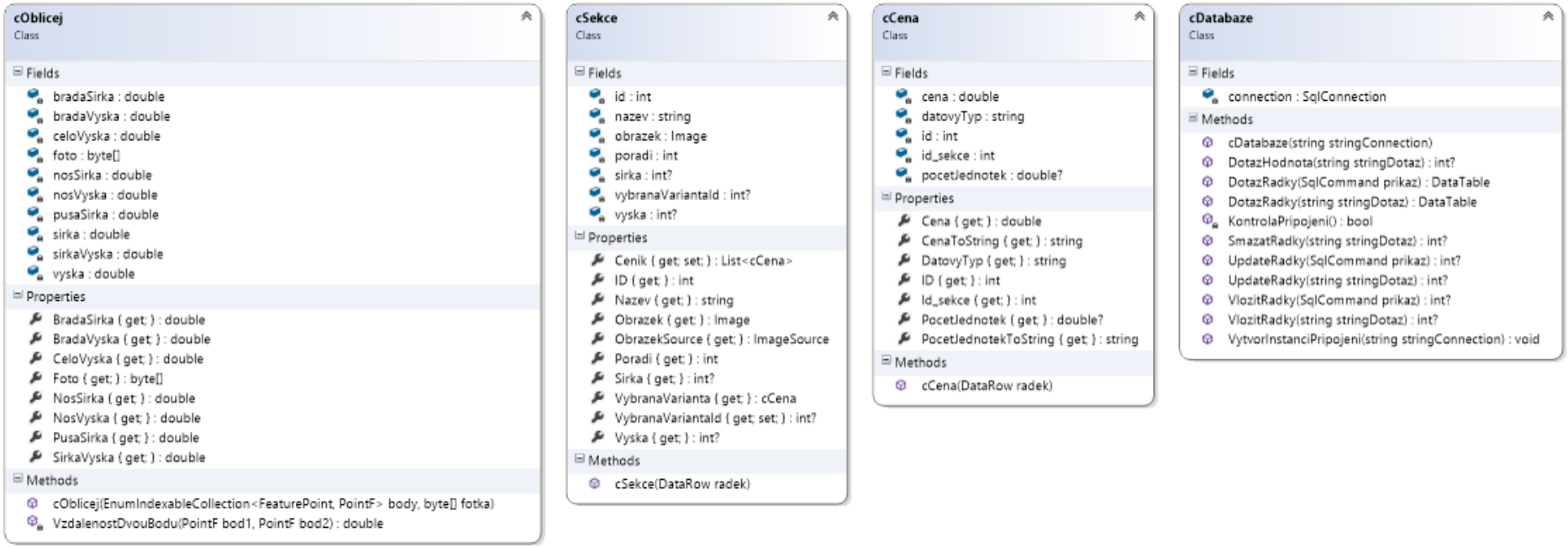
Webb, J., & Ashley, J. (2012). *Beginning Kinect programming with the Microsoft Kinect SDK*. (p. xvii, 306 p.). New York: Apress.

III Seznam obrázků

IV Seznam zdrojových kódů

V Seznam příloh

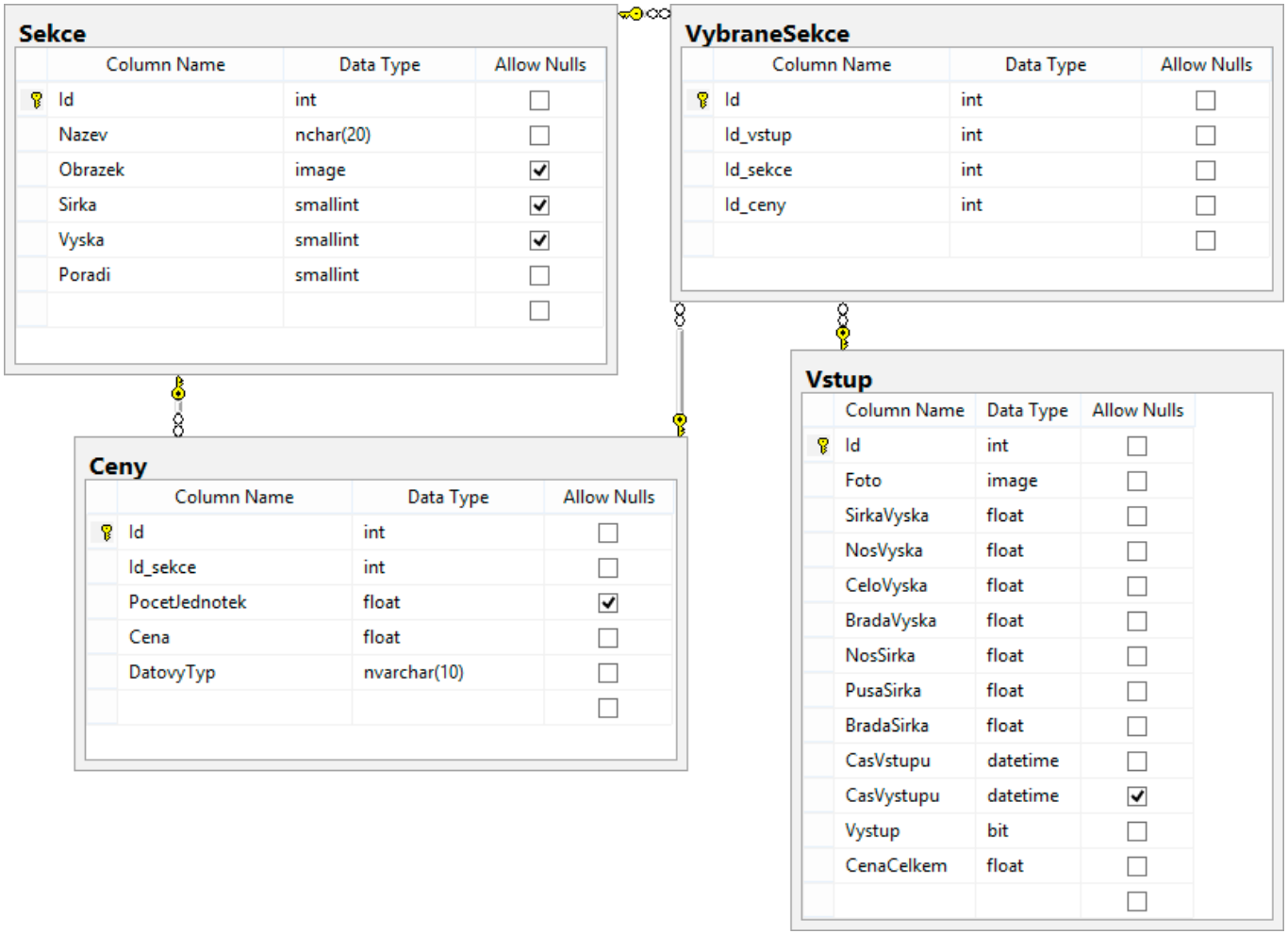
VI Přílohy



Příloha A Diagram tříd

	Šířka obličeje / výška obličeje	Výška nosu / výška obličeje	Výška čela / výška obličeje	Výška brady / výška obličeje	Šířka nosu / šířka obličeje	Šířka pusy / šířka obličeje	Šířka brady / šířka obličeje
Test 1	0,6470847	0,2804566	0,3001850	0,1394473	0,1703913	0,2427475	0,2665203
Test 2	0,6458626	0,2800852	0,3089365	0,1396492	0,1718821	0,2581224	0,2687395
Test 3	0,6429847	0,2850391	0,3007048	0,1428803	0,1742741	0,2728763	0,2735615
Odchylka	**8,862E-06**	**1,523E-05**	**4,821E-05**	**7,422E-06**	**7,673E-06**	**4,539E-04**	**2,592E-05**

Příloha B Tabulka porovnání relativních vzdáleností obličeje v ideálních podmínkách



Příloha C Databázové schéma