Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Bakalářská práce

SAGElab - Bezdotykové ovládání

Vedoucí práce: Ing. Jiří Melnikov

5. května 2015

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce, panu Ing. Jiřímu Melnikovi, za cenné rady a čas, který mi věnoval při konzultacích bakalářské práce. Dále děkuji své rodině za neustálou podporu při mém studiu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 5. května 2015 .....

Odkaz na tuto práci

Sedláček, Lukáš. *SAGElab - Bezdotykové ovládání*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Práce se zaměřuje na problém bezdotykového ovládání vizualizačního prostředí SAGE2. Hlavním tématem je analýza dostupných řešení bezdotykového ovládání telestěny a zařízení určených ke snímání pohybu. Rešerše také obsahuje návrh vlastní aplikace řešící zadaný problém. Cílem bakalářské práce je tedy návrh a implementace prototypu aplikace, který zprostředkuje bezdotykové ovládání prostředí SAGE2 pomocí výše zmíněného zařízení. Závěrem jsou poznatky získané z uživatelského testování aplikace a doporučení pro další rozvoj aplikace.

Klíčová slova Bezdotykové ovládání, telestěna, SAGElab, vizualizační prostředí SAGE2, gesta rukou, hloubkový snímek, zařízení snímající pohyb

Abstract

The thesis is focused on a contactless controlling of a visualization environment SAGE2. The main topic is an analysis of available solutions of contactless controlling and of gesture based devices. A research also contains a design of an own solution which is resolving given issue. The goal of the bachelor's thesis

is therefore a design and implementation of an application which conveys contactless controlling of the SAGE2 environment using the aforementioned device. The conclusion contains findings which were acquired during application testing and a recommendation for further development.

Keywords Contactless controller, wallscreen, SAGElab, Scalable amplified graphics environment, hand gestures, depth frame, motion detection device

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době se stále vyvíjejí nové a inovativnější způsoby jak ovládat počítače. Některé způsoby, jako například hlasové ovládání počítače, mohou usnadnit lidem s tělesným či jiným postižením práci s těmito technologiemi. Jiné způsoby jsou vyvíjeny pro zjednodušení ovládání prezentací či vizualizací. Nejvíce nových druhů ovládání se rozvíjí v zábavním průmyslu. V souladu s těmito fakty existuje již velké množství dotykových technologií pro ovládání počítačů. Postupně se ale začínají vyvíjet nové možnosti založené na bezdotykové interakci s počítači.

Tématem bakalářské práce je především seznámení se s již existujícími možnostmi bezdotykového ovládání a navržení nového vlastního řešení, pomocí kterého bude možno ovládat telestěnu s vizualizačním prostředím Scalable Amplified Group Environment (SAGE2). Řešení by mělo být schopno nahradit činnosti prováděné počítačovou myší a mělo by být dobře využitelné v laboratoři SAGElab. Literární rešerše se věnuje analýze existujících řešení, způsobu vlastního řešení a testování prototypu. Praktická část je zaměřena na samotnou implementaci nově navrhnutého prototypu aplikace, která podporuje bezdotykové ovládání kuzoru zmíněného prostředí.

Kapitola 1

Vysvětlení pojmů

Aby bylo možné plně porozumět následujícímu textu bakalářské práce, je nutné vysvětlit objevující se nestandardní pojmy tak, aby i neznalý čtenář neměl při čtení bakalářské práce problémy s pochopením textu. Kapitola popisuje nečastěji používané klíčové pojmy v práci.

SAGE SAGE je vizualizační prostředí, které umožňuje zobrazení vizualizací, animací a dalších informací na telestěnách ve vysokém rozlišení. Cílem při implementaci prostředí bylo zobrazení velkého množství dat ve velké rychlosti, přesnosti a především ve vysokém rozlišení na telestěně. Prostředí je vyvíjeno laboratoří EVL[1] a zdrojové kódy implementace jsou veřejně dostupné.

Prostředí je instalováno v laboratoři SAGElab[2] a využívá se zejména k prezentačním účelům a videokonferencím. V laboratoři jsou dostupné verze SAGE1 a SAGE2. Poháněno je v současnosti šesti výkonnými servery a využívá telestěnu složenou z dvaceti obrazovek celkově s úlopříčkou 5,58 metrů (rozlišení telestěny je 9600x4320 pixelů).

SAGE1 Scalable Adaptive Graphics Environment (SAGE1)[3] je původní verze vizualizačního prostředí implementovaná v jazyku C++. Pro zobrazení základního uživatelského rozhraní jsou použity skripty v jazyku Python (zobrazení spuštěných oken, reprezentace kurzoru). Prostředí je spustitelné pouze na linuxových operačních systémech a vyžaduje jistou znalost správy těchto systému (ovládání telestěny a spouštění aplikací je problémové pro méně zkušeného uživatele). SAGE1 umožňuje zobrazování multimédií ve vysokém rozlišení, sdílení vlastní plochy do prostředí, apod. Využití prostředí na telestěně při prezentaci lze vidět na obrázku 1.1.

SAGE2 Scalable Amplified Group Environment (SAGE2)[4] je pokročilejší verzí původního vizualizačního prostředí, která je implementovaná v programovacím jazyku Javascript. Prozatím je prostředí dostupné pouze v beta

Obrázek 1.1: Prezentace s použitím prostředí SAGE1, obrázek převzat z webu[2]

verzi. Na rozdíl od původního prostředí SAGE1 je novější verze multiplatformní, protože základem jsou webové technologie a programovací jazyk Javascript. Zároveň bylo vylepšeno uživatelské rozhraní, které je nyní intuitivní a prostředí tak může ovládat i méně zkušený uživatel (na obrázku 1.2 lze vidět ovládání prostředí kurzorem). Uživatelské rozhraní je spuštěno v internetovém prohlížeči a lze jej ovládat (i vzdáleně) z jakéhokoli zařízení (mobilní telefon, tablet, laptop).

SAGE pointer *SAGE pointer* je reprezentace klasického počítačového kurzoru ve vizualizačním prostředí SAGE. Události kurzoru (pohyb, klikání, pohyb kolečkem) jsou mapovány na události vstupního zařízení. Základem ovládání *SAGE pointeru* je tedy aplikace, která umí přemapovat události ze vstupního zařízení na událostí kurzoru. Vstupem nemusí být pouze počítačová myš, avšak může jím být jakékoli jiné zařízení, pro které je implementovaná mapovací aplikace (zmíněný typ aplikace je také předmětem bakalářské práce). V textu se také pojem vyskytuje pod názvem „kurzor prostředí SAGE“.

Hloubkový snímek Často se tento pojem nepřekládá a lze jej najít pod anglickým názvem „depth frame“. Hloubkový snímek je snímán zařízením s hloubkovým senzorem a obsahuje informace o vzdálenostech objektů od senzoru. Z hlediska typu uchovávaných jsou snímky často základem aplikací bez-

Obrázek 1.2: Prostředí SAGE2

dotykového ovládání. Nejčastěji se zobrazují ve stupních šedi jako lze vidět na obrázku 1.3.

Hloubkové zařízení/kamera Hloubkové zařízení obsahuje senzory pro snímání hloubkových snímků. Tato zařízení se často využívají k bezdotykovému ovládání.

Knihovna Programovací knihovna je soubor funkcí, objektů, vlastních datových typů a dalších částí zdrojových kódů, které mohou být sdíleny více aplikacemi. Knihovny se používají ke zjednodušení implementace aplikace, která poté komunikuje s knihovnou přes její aplikační rozhraní[1]. Knihoven existuje velké množství s různými účely, např.: grafické knihovny, knihovny pro správu připojených zařízení, serverové knihovny apod.

Websocket Protokol websocket[6] umožňuje odesílání zpráv po vytvořeném TCP spojení oběma směry zároveň, tzn. server může odesílat data klientovi nezávisle na tom, zda klient data vyžádal. Websockety byly navrženy především pro webové aplikace a webové prohlížeče. **Websocket server** vytvoří spojení a čeká na připojení klientů. **Websocket klient** se připojuje na server nejčastěji pomocí URL adresy serveru.

[1] funkce, třídy nebo procedury knihovny dostupné pro použití programátorem

Obrázek 1.3: Hloubkový snímek[5]

Kapitola 2

Současná řešení

Kapitola popisuje a analyzuje současná existující řešení problému bezdotykového ovládání. Zaměřuje se především na dostupná řešení pro původní vizualizační prostředí SAGE1, jelikož prostředí SAGE2 je velice mladé (pro veřejnost zpřístupněno 24. listopadu 2014[4]) a neexistují zatím žádné volně dostupné aplikace, které by bezdotykové ovládání umožňovaly. Dále kapitola popisuje další zajímavé moderní technologie bezdotykového ovládání.

2.1 Řešení pro SAGE

2.1.1 Základní informace

Aplikace bezdotykového ovládání pro původní prostředí je implementována laboratoří EVL, která taktéž implementovala prostředí SAGE1 a SAGE2. Řešení využívá *SAGE pointer* a TCP spojení se serverem SAGE1, po kterém se posílají všechny události, které jsou předávány kurzoru prostředí.

Implementace sleduje pohyby pouze jedné ruky, tzn. mapuje pohyby ruky na pohyby kurzoru a zároveň sleduje, zda stejná ruka neudělala gesto[7]. Způsob nemusí být vždy vhodný, jelikož pokud se provádí gesto pro kliknutí na určitý prvek zobrazený v prostředí, může se při provádění gesta hýbnout s kurzorem a ve výsledku kliknout na místo jiné, než se původně zamýšlelo. Zároveň použití pouze jedné ruky omezuje možnosti detekovaných gest.

Samotné implementované řešení od laboratoře EVL není volně k dispozici, ovšem pro programátory je volně dostupná knihovna omicron[8] od stejnojmenné laboratoře. Z oficiálních webových stránek EVL se odkazuje právě na omicron, při vyhledávání bezdotykové aplikace, lze tedy předpokládat, že je řešení na knihovně postaveno. Knihovna může být integrována do C++ aplikací a s její pomocí je možno implementovat řešení bezdotykového ovládání.

2.1.2 Kompatibilní zařízení

Za předpokladu, že je použita zmíněná knihovna omicron, je možné k aplikaci připojit velké množství zařízení různých druhů. Avšak jediné hloubkové zařízení určené pro bezdotykovou interakci (tento typ zařízení bude použit v praktické části), které je možné k řešení připojit, je Microsoft Kinect (viz. 3.1.1). Jiné alternativy hloubkových senzorů nejsou podporovány a nejsou řešením rozpoznány.

2.2 Další řešení

Kromě způsobů využívající technologii hloubkových snímků (která je předmětem bakalářské práce a je popisována v dalších kapitolách), existují i další technologie bezdotykového ovládání. Mezi nejzajímavější (dle autorova názoru) patří ovládání pomocí snímání gest dlaní a prstů a ovládání pomocí prstenů.

2.2.1 Leap Motion

Leap Motion[9] je zařízení (nejčastěji ležící na stole), které umožňuje bezdotykové ovládání prostředí, her a aplikací pomocí rukou. Řešení umí detekovat pozice dlaní a gesta prováděná rukou (zejména pomocí prstů). K počítači je zařízení připojeno pomocí USB. Leap Motion umí sledovat obě ruce uživatele, což přidává další možnosti použití. Pro zařízení jsou prozatím dostupné hry a aplikace pouze na platformách Mac OSX a Microsoft Windows. Ovšem pro vývojáře je dostupný *Software development kit*[2], který je možné instalovat i na platformě Linux.

2.2.2 Ovládání prsteny

Bezdotykové ovládání pomocí prstenů je velice mladá technologie. Z toho důvodu jsou dvě ze tří zmíněných zařízení prozatím dostupná běžným zákazníkům pouze v předprodeji.

iRing Pohybový ovladač iRing[10] je řešení bezdotykového ovládání hudebních aplikací a efektů na zařízeních od společnosti Apple, Inc. Aplikace podporující iRing jsou tedy dostupné pouze na platformě iOS. Pomocí kamery chytrého telefonu či tabletu je sledována poloha prstenu. K ovládání je možné použít více prstenů. Řešení umí detekovat různá gesta (rychlé přiblížení k senzoru, náklon prstenu či zobrazení a schování prstenu). Zároveň za pomoci Wi-Fi připojení je možné ovládat aplikace z Mac OSX.

[2]Knihovna od výrobce, která umožňuje programovat aplikace pro zařízení.

Ring Na rozdíl od prstenu iRing bude použití Ring[11] multiplatformní. Prozatím existují aplikace pro platformu iOS, avšak vyvíjí se i aplikace pro mobilní operační systém Android. Je to také zařízení, které je prozatím pro běžné zákazníky v předprodeji (do 30. dubna 2015). Funkčnost Ring je založena na spárování s mobilním zařízením (chytrým telefonem, tabletem) pomocí Bluetooth[3] nebo Wi-Fi připojení. Mobilní zařízení poté ovládá jiné spotřebiče v domácnosti pomocí infračerveného portu či Wi-Fi spojení. Oproti iRing nejsou gesta prstenu sledována po celou dobu práce s prstenem, avšak detekce gest započne po stisknutí příslušného tlačítka na prstenu.

Nod Použití prstenu Nod[12] je velice podobné předchozímu prstenu Ring. Ovšem na rozdíl od Ring je možné využít více prstenů a tím docílit velmi přesného sledování pohybu horních končetin uživatele. Prsteny Nod by měly být dobře použitelné zejména pro hraní her. Použití by mělo být dostupné pro platformy Windows, OSX, Linux, Google Glass, iOS a Android. Nod stejně jako předchozí zařízení je prozatím jen v předprodeji a pro běžné uživatele není dostupný.

[3]Spojení mezi zařízeními na krátké vzdálenosti

Kapitola 3

Analýza

Analýza je prvním krokem při tvorbě nového software. Dává návrhářům aplikace určitou představu o výsledném produktu. Pomocí dobře provedené analýzy můžeme snadno odhalit určitá rizika, která by mohla vzniknout při pozdějším návrhu nové aplikace. Výsledkem analýzy by měla být zpráva určující, kdo bude program používat, co všechno musí umět a další rozhodnutí důležitá pro návrh aplikace.

V kapitole jsou nejprve analyzována zařízení pro snímání pohybu a pohybových gest (se zaměřením na zařízení podporující hloubkové snímání). Vybraná zařízení jsou nejznámější v oblasti snímání pohybových gest uživatele. Dále jsou popsány požadavky, které jsou na prototyp aplikace kladeny a případy užití, které definují přesné interakce uživatele s aplikací. V kapitole se také vyskytuje sekce, která ověřuje nezbytnost všech funkčních požadavků kladených na systém. Sekce je důležitá zejména k určení chyb v analýze, kdy vznikají zbytečné funkční požadavky nebo případy užití.

3.1 Zařízení pro snímání pohybu

Aby bylo možné ovládat prostředí bezdotykově, je nutné použít vhodné zařízení, pomocí kterého bude možné určit pozice ruky a provedená gesta. Analyzovaná zařízení jsou nejznámější a často nejpoužívanější v oblasti snímání pohybových gest. Všechna vybraná zařízení obsahují senzor pro snímání hloubkových snímků. Detekování pohybových gest uživatele poté vychází ze zpracovaných hloubkových snímků.

3.1.1 Kinect

Jedním z nejstarších dostupných zařízení pro bezdotykové ovládání je Microsoft Kinect. Často jej lze vidět ve spojení s herními konzolemi Xbox[13]. Zařízení má velkou podporu na všech operačních systémech, proto je stále nejpoužívanější i přes fakt, že hloubková kamera má velice malé rozlišení. Existuje

Tabulka 3.1: Porovnání zařízení

	Kinect	Kinect v2	Asus Xtion
Rozlišení hloubkové kamery	320x240	512x424	320x240
Ovladače	Windows Linux Mac OS	Windows	Windows Linux Mac OS
Knihovny	Velké množství open-source knihoven založené na OpenNI, NiTE a OpenKinect knihovnách	Kinect SDK2.0	OpenNI NiTE

také alternativa Kinect for Windows, která je použitelná především pro stolní počítače. Na počítačích s operačním systémem Linux je paradoxně lepší využívat Kinect ve verzi Xbox, jelikož dostupné ovladače jsou s tímto zařízením stabilnější než s verzí pro Windows.

3.1.2 Kinect v2

Od roku 2014 je k dispozici druhá verze zařízení Kinect[14] (ve verzi pro Xbox i Windows). Druhá generace již zaznamenává hloubkové snímky ve vyšším rozlišení. Vzhledem k mládí zařízení jsou ovladače a knihovny dostupné pouze pro operační systém Windows a zařízení prozatím není podporováno na alternativních operačních systémech.

3.1.3 Asus Xtion PRO

Asus Xtion PRO[15] je alternativou ke staršímu zařízení Kinect (viz. 3.1.1). Obdobně jako původní verze Kinect, tak i zařízení Xtion snímá ve stejně nízkém rozlišení a je pro něj poměrně velká podpora na všech operačních systémech. Zařízení je menší než Kinect a na rozdíl od Kinectu je napájeno pouze pomocí USB. Xtion ve verzi PRO obsahuje pouze hloubkový senzor, ve verzi PRO Live poté obsahuje stejné senzory jako Kinect.

3.1.4 Porovnání

Při analýze je nutné porovnat dostupná zařízení, aby byly zřejmé možné výhody a nevýhody analyzovaných zařízení. Na základě analýzy zařízení se později vybere vhodné zařízení pro implementaci prototypu aplikace. V tabulce 3.1 jsou ovladače porovnány v parametrech důležitých pro správné a přesné fungování prototypu. Rozlišení jednotlivých zařízení obsažených v tabulce jsou dohledatelné z referencí [16][14][15], další údaje vychází z vlastního průzkumu dostupných ovladačů a knihoven.

3.2 Požadavky

Sekce popisuje funkční a nefunkční požadavky kladené na aplikaci. Analýza požadavků je z části výsledkem společných schůzek bakalantů, jejichž téma bakalářské práce se také zaměřuje na prostředí SAGE2. Nefunkční požadavky (viz. 3.2.1) vychází z implementace prostředí a technologie bezdotykového ovládání. Funkční požadavky (viz. 3.2.2) jsou definovány dle možností ovládání prostředí.

3.2.1 Nefunkční požadavky

- **N1 Linuxová architektura** Aplikace musí být spustitelná na linuxových 64bitových operačních systémech. Důvodem je především kompatibilita na počítačích v laboratoři SAGElab. Konkrétní distribuce linuxového operačního systému nebude aplikací omezena.
- **N2 C++ programovací jazyk** Aplikace musí být napsaná v jazyku C++. Jazyk je zvolen s ohledem na dostupné knihovny, které jsou převážně v jazyku C/C++. S javascriptovým prostředím SAGE2 bude komunikovat pomocí websocketů zprávami ve formátu JSON (viz. 4.1).
- **N3 Dostupnost k síti** K běhu aplikace musí být správně nakonfigurované síťové připojení a dostupnost k síti, aby bylo umožněno připojení k SAGE2 serveru a komunikace pomocí websocketů.
- **N4 Libovolné hloubkové zařízení** Bude možnost připojit jakékoli zařízení, které má zabudované hloubkovou kameru (Asus Xtion, Xbox Kinect, Kinect for Windows) a pro které existují dostupné ovladače na daném operačním systému.
- **N5 Konfigurovatelnost aplikace** Program musí umět načítat data z konfiguračního souboru, aby bylo možné aplikaci uživatelsky nastavit. V nastavení musí být především IP adresa a port serveru, rozlišení výchozí obrazovky či telestěny a další hodnoty související se samotným snímáním pohybu.

3.2.2 Funkční požadavky

- **F1 Mapování pohybů ruky na pohyb kurzoru** Aplikace musí umožnit mapování pohybů ruky na pohyb kurzoru ve vizualizačním prostředí. Při mapování je důležité, aby se kurzor ve všech směrech pohyboval stejnou rychlostí nezávisle na poměru stran telestěny.
- **F2 Detekce gest** Krom správného detekování pohybů ruky je také nutné detekovat různá gesta rukou a to tak, aby byla nezaměnitelná mezi sebou a aby byla umožna umožněna plnohodnotná správa oken. Pro plné využití možností prostředí definujeme gesta simulující:

- Klik pravým tlačítkem myši
- Klik levým tlačítkem myši
- Klik prostředním tlačítkem myši
- Pohyb kolečkem myši

3.3 Případy užití

Případy užití (často také nazýváno *use-case*) je definovaný seznam interakcí mezi uživatelem a aplikací. Use-case by měly pokrýt všechny funkční požadavky (viz. 3.2.2) kladené na systém (v případě, že se tak nestane, je nutné se zamyslet nad nutností všech funkčních požadavků).

Modelování případů užití[17] se skládá ze:

- Seznamu účastníků
- Diagramů případů užití
- Seznamu případů užití

3.3.1 Seznam účastníků

Z povahy práce vyplývá, že existuje pouze jediný účastník aplikace - uživatel. Uživatel má možnost manipulovat pomocí gest s okny v prostředí SAGE2 stejně, jako kdyby prostředí ovládal počítačovou myší. Zároveň je uživateli umožněno měnit nastavení aplikace pomocí konfiguračního souboru.

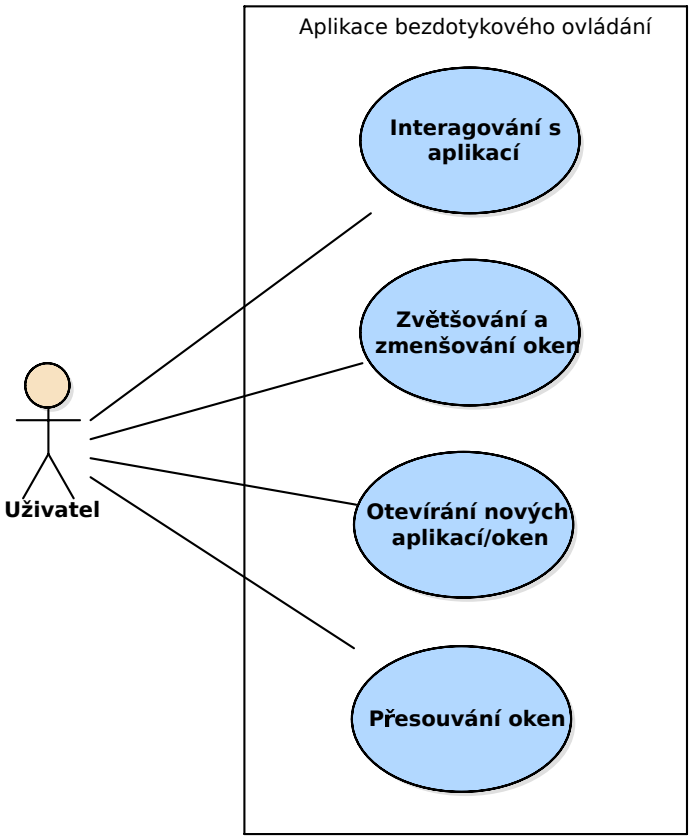
3.3.2 Diagram případů užití

Hlavním účelem diagramu je zachycení interakce mezi účastníkem systému a samotnou aplikací, tzn. popisuje vztahy mezi případy užití a účastníky. Diagram případů užití je zachycen na obrázku 3.1.

3.3.3 Seznam případů užití

Seznam případů užití je důležitou a nedílnou součástí analytického modelování. V sekci jsou podrobně popsány a definovány případy užití tak, aby uživatel mohl plně bezdotykově ovládat vizualizační prostředí SAGE2.

- **UC1 Přesouvání oken** Uživatel má možnost libovolně přesouvat spuštěná okna ve vizualizačním prostředí. Uchycení oken je umožněno pomocí gesta pro simulaci kliku levým tlačítkem myši a přesunutí okna je poté umožněno pomocí kurzoru namapovaného na pohyb ruky.

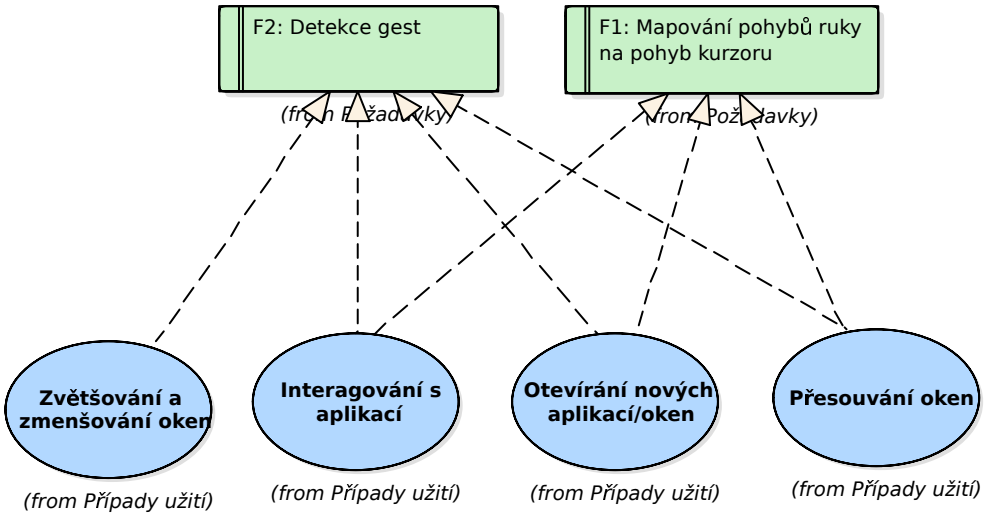


Obrázek 3.1: Diagram případů užití

- **UC2 Otevírání nových aplikací/oken** Aby uživatel mohl plnohodnotně pracovat s telestěnou, musí mít možnost otevírat nová okna aplikací. Aplikace je možné spouštět z vizualizačního prostředí přes menu vyvolané událostí pravého kliku myši.

- **UC3 Zvětšování a zmenšování oken** Dává uživateli možnost zvětšovat a zmenšovat jednotlivá otevřená okna, která jsou v danou chvíli aktivní. Aktivním oknem se rozumí takové okno, na kterém je přesunutý kurzor.

- **UC4 Interakce s aplikací** Umožňuje přepínat mezi módy „manipulace s okny“ a „interakce s aplikací“ a následně interagovat s konkrétní vybranou aplikací spuštěnou v prostředí. Při zapnutém módu „interakce s aplikací“ není možné nijak manipulovat se spuštěnými okny. Uživatel musí mít možnost přepínat mezi módy pomocí definovaných gest.



Obrázek 3.2: Mapování požadavků na případy užití

3.4 Ověření nezbytnosti požadavků

Při vytváření nové aplikace je důležité ověřit, zda jsou všechny funkční požadavky zahrnutné v některém z případů užití. Pokud by se stalo, že by existoval požadavek, který není zahrnutý v žádném případu užití, pak je nutné se zamyslet na důležitostí daného požadavku či nad chybějícím případem užití.

Z obrázku 3.2 lze vidět, že všechny funkční požadavky jsou namapovány na případy užití, tzn. nejsou definovány žádné zbytečné funkční požadavky.

Kapitola 4

Návrh

Po analytické části (viz. 3) je dalším krokem při vývoji software návrh aplikace. Při návrhu aplikace se již zabýváme konkrétními technologiemi (jazyk programu, formát odesílaných zpráv na server, formát konfiguračního souboru, dostupné knihovny) a také architekturou počítače, na kterém bude možné aplikaci spustit. Cílem návrhu je promyšlení implementace aplikace a navržení jednoduchého a efektivního řešení, ve kterém je většina rizikových částí implementace vyřešená.

V kapitole je dále popsáno rozdělení aplikace, které velký problém aplikace bezdotykového ovládání rozkládá na menší a lépe řešitelné problémy, které budou snadněji programovatelné. Při implementaci bude nutné využít knihovny, které zjednodušší řešení. Knihovny budou použity pro části praktické práce, které nejsou hlavním předmětem zadání. V kapitole je popsaná většina dostupných knihoven a dále knihovny, které budou v práci použity. Z popisu návrhu a předchozích kapitol by mělo být zřejmé, jak aplikaci implementovat a také na jakém prostředí ji spouštět.

4.1 Jazyk programu

Programovací jazyk je dle nefunkčních požadavků 3.2.1 definován jako C++. V tomto jazyku je implementována i většina dostupných knihoven. C++ jazyk je také nejvhodnější vzhledem k povaze aplikace, která musí zpracovat 30-60 snímků za sekundu, jelikož bývá rychlý i na méně výkonných počítačích. Zároveň jazyk umožňuje objektově orientované programování, které se stalo normou vývoje aplikací.

Zprávy, které bude aplikace posílat SAGE2 serveru, musí být ve formátu JSON[18]. Formát je nutno striktně dodržet, aby mohly být zprávy správně zpracovány serverem. Příklad možných JSON zpráv posílaných serveru:

```
{"f":"startSagePointer"}
{"f":"pointerPress","d":{"button":"right"}}
```

```
{"f":"ptm","d":{"dx":30, "dy":20}}
```

Jak lze vidět, zpráva je často složená ze dvou částí. První „f“ část obsahuje název příkazu. Všechny možné příkazy použitelné k ovládání kurzoru prostředí jsou popsány v sekci 5.3.2. Druhá „d“ část obsahuje pole dat příkazu (počet pixelů o který se kurzor posouvá, tlačítko které se mačká, apod.).

4.2 Konfigurace aplikace

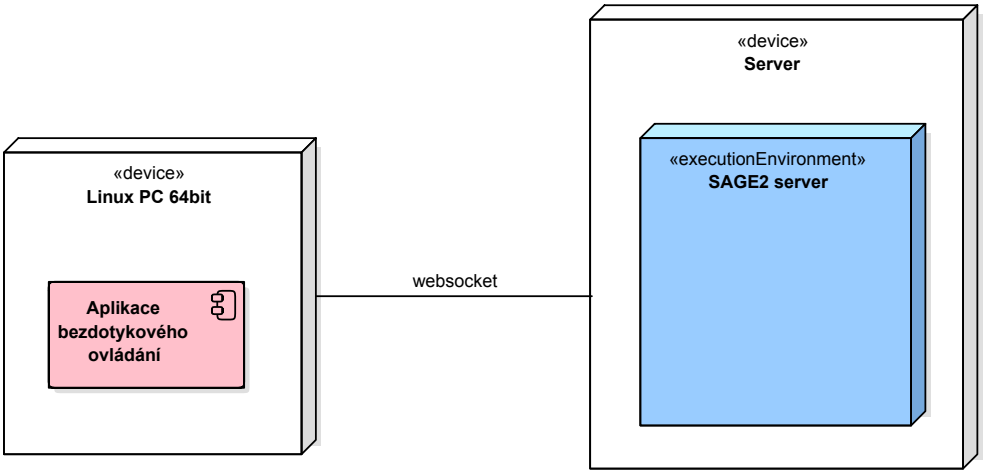
Zároveň je také nutné definovat přesný formát konfiguračního souboru, ze kterého budou po startu aplikace načtené základní proměnné aplikace. Pomocí konfiguračního souboru musí být možné nastavit přípojný bod aplikace (síť), rozlišení a jiné hodnoty výstupní obrazovky a hodnoty měnící nastavení trackovacích algoritmů[4]. Knihovny, které budou v práci použity, používají formát konfiguračních souborů INI[19], tudíž je vhodné dodržet stejný formát souborů v praktické části práce. Zároveň zmíněný formát patří ke standardům pro ukládání inicializačních hodnot aplikace. Příklad INI konfiguračních souborů:

```
;Comment
[Header1]
attribute1=value1
attribute2=value2
...
```

4.3 Model nasazení

Podobně jako jazyk programu i architektura systému, na kterém bude možno aplikaci pro bezdotykové ovládání spouštět, je definována z nefunkčních požadavků 3.2.1. Operační systém tedy musí být 64bitový linuxový systém. Omezení na linuxový systém vychází z možností počítačů v laboratoři SAGElab. Zároveň dle operačního systému jsou zvoleny i dostupné knihovny. Z knihoven poté vyplývá, že aplikace nebude přenositelná na jiné operační systémy bez vzniklých problémů kompatibility. Na obrázku 4.1 je diagram nasazení znázorněn.

[4] Algoritmy sledující pohyb uživatele.



Obrázek 4.1: Diagram nasazení

4.4 Rozdělení aplikace

Pro lepší návrh a pozdější implementaci je vhodné aplikaci rozdělit na části. Každá část aplikace se poté zaměřuje pouze na jeden konkrétní problém, který je řešen implementační třídou nebo skupinou tříd a funkcí. Přesné popisy těchto implementovaných tříd a metod jsou definovány v kapitole 6. Problémy, dle kterých je aplikace rozdělenena, jsou:

- Načtení konfiguračního souboru
- Inicialiazace knihoven a zařízení
- Websocket klient pro odesílání zpráv serveru
- Mapování pohybových gest na události kurzoru prostředí

4.4.1 Načtení konfiguračního souboru

Před inicializací aplikace bude nutné načíst hodnoty definované v konfiguračním souboru do aplikace. Část aplikace, která načítání ze souboru provede, bude muset umět parsovat konfigurační soubory formátu INI, kontrolovat jejich validitu a připravit hodnoty pro použití v aplikaci.

4.4.2 Inicializace knihoven a zařízení

Inicializace se musí provést po každém spuštění aplikace. Některé hodnoty nutné pro inicializaci jsou načtené z konfiguračního souboru. Při inicializaci se připraví použité knihovny a také zařízení s hloubkovým senzorem, které je k počítači připojeno.

4.4.3 Websocket klient

Klientská část aplikace má za úkol připojit se při inicializaci na websocket server poskytovaný SAGE2 prostředím. K připojení využije hodnoty načtené z konfiguračního souboru nebo základní hodnoty. Po připojení bude odesílat zprávy ve formátu JSON na websocket server.

4.4.4 Mapování pohybových gest na události kurzoru prostředí

Samotné jádro aplikace bude tvořeno mapováním pohybových gest rukou na události *SAGE pointeru*. K ovládání prostředí budou využity obě ruce. Jedna ruka bude mapována na pohyby kurzoru v prostředí. Pohyb druhé ruky bude sledován a při případných zachycených gestech budou pohyby přemapovány na události *SAGE pointeru*, které budou odeslány na server.

4.5 Dostupné knihovny

V implementační části je nutné se zaměřit především na algoritmy pro detekování gest z hloubkových snímků. Ovšem v praktické části jsou i jiné komplexní či algoritmicky složité části, které nejsou hlavním předmětem řešení praktické části (práce s připojeným zařízením, extrahování dat ze snímků, atd.). Při návrhu je nutné vybrat knihovny, které zjednoduší implementaci, tzn. knihovny, které budou pracovat se:

- Zařízením a zahrnující otevření a zavření připojeného zařízení a získávání hloubkových snímků ze zařízení
- Hloubkovými snímky, tzn. extrakce důležitých informací z těchto smínků
- Spojením se serverem a odesíláním zpráv pomocí websocketů

V praktické části budou použity další standardní C a C++ knihovny, které podpoří řešení. Standardní knihovny jsou nainstalovány v operačním systému spolu s kompilátorem, tudíž není potřebné je instalovat zvlášť. Knihovny použité v práci nebudou používat žádné prvky C++11.

4.5.1 Knihovna pro zařízení

Z nefunkčních požadavků (viz. 3.2.1) je zřejmé, že aplikace musí umět připojit více typů zařízení. Není tedy vhodné používat nejrozšířenější knihovnu pro práci s Kinect (viz. 3.1.1) OpenKinect[20] z důvodu možnosti připojení pouze jednoho typu zařízení. Zároveň by bylo složité používat pro každé zařízení jinou knihovnu, tudíž je vhodné použít jeden typ knihovny, který zvládne ovládat různé typy zařízení s hloubkovým seznzorem.

Nejvhodnější knihovna pro praktickou část se nazývá OpenNI2 a je vyvíjena společností Structure.io[21]. Společnost se zaměřuje především na práci se zařízeními s hloubkovým senzorem, která jsou pro aplikaci bezdotykového ovládání vhodná, tudíž knihovna podporuje velké množství právě těchto zařízení. V knihovně jsou také implementované ovladače pro podporovaná zařízení, které fungují se všemi typy až na ovladač Kinect. Ovšem pro Kinect lze využít ovladačů dodávaných již zmíněnou knihovnou OpenKinect a tím zajistit plnou funkčnost zařízení i s knihovnou OpenNI2.

4.5.2 Knihovna pro zpracování snímků

Existuje velké množství knihoven, které umí zpracovávat obraz. Mezi nejznámější patří knihovna OpenCV[22], která poskytuje nemalé množství funkcí pro práci s obrazovými daty. Ovšem pro extrakci pozic rukou z hloubkových snímků není příliš vhodná, jelikož by řešení bylo algoritmicky složité. Lepším řešením je využít knihovnu NiTE2 vyvíjenou společností PrimeSense[23], která má již implementované algoritmy pro extrakci důležitých bodů ze snímků.

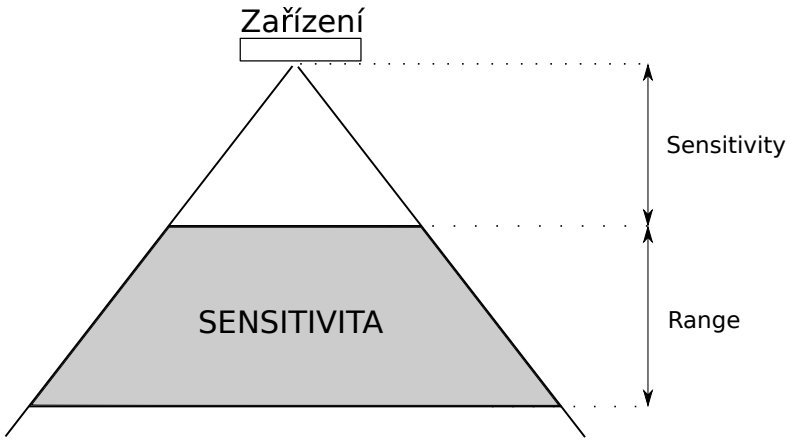
Knihovna NiTE2 je vázána na použití OpenNI2 zmíněné v předchozí sekci 4.5.1. Hloubkové snímky získané z připojeného zařízení jsou zpracovány a pomocí knihovních algoritmů jsou extrahovány důležité informace ze snímků. Knihovna se umí zaměřit na celou osobu a sledovat jeji pohyby[5], nebo sleduje pouze pohyby rukou [6]. V závislosti na použitých algoritmech jsou poté extrahovanými informacemi pozice kloubků celé osoby nebo pouze pozice rukou. Rozdílem mezi algoritmy je především větší přesnost při použití algoritmu sledování rukou. Pozice jsou dány ve světových souřadnicích zařízení s měřítkem v milimetrech odpovídající skutečným vzdálenostem od os senzoru.

Knihovna OpenCV se může použít k vylepšení snímků získaných ze zařízení před samotnou extrakcí informací (vzniklá gesta, počet rukou, apod.) pomocí NiTE2. Vylepšením se rozumí jakékoli postprodukční úpravy obrazu, které povedou k lepším výsledkům při extrakci informací z obrazu. V praktické části nebude OpenCV použita z důvodu zachování jednoduchosti implementace.

Problém s knihovnou Při testování knihovny NiTE2 se bohužel dospělo k závěru, že gesta detekovatelná danou verzí knihovny nefungují. Dle názoru vedoucího bakalářské práce se problém vyskytuje pouze s použitím 64bitové verze knihovny. Implementovaná gesta měla být využita především při prvotní detekci rukou, aby bylo možné zaměřit přesný pár rukou, který bude aplikací snímán (například máváním či jiným gestem se zaměří ruka a její inicializační pozice v prostoru).

Problém knihovny bude vyřešen tak, že první dvě načtené ruce budou sledovány. Na ostatní ruce zaměřené před kamerou nebude brán zřetel a nebudou

[5] Časteji je proces k nalezení pod pojmem skeleton tracking.
[6] K nalezení pod pojmem hand tracking



Obrázek 4.2: Nastavení senzitivity zařízení

aplikací sledovány. Zároveň se nastaví senzitivita zařízení (obrázek 4.2; vzdálenosti jsou popsané atributy z konfiguračního souboru), aby uživatel mohl být zaměřen pouze v určité vzdálenosti od senzoru. Pomocí těchto dvou omezení (první načtené ruce a senzitivita) by se měl přesněji zaměřit pár rukou jednoho uživatele, který bude aplikací sledován.

4.5.3 Knihovna pro práci s websockety

SAGE2 je websocketový server, který čeká na připojení klientů. Klienti pomocí websocketů naváží spojení se serverem, přes které následně odesílají zprávy. Zprávy mohou inicializovat nastavení serveru vzhledem k připojenému klientu, nebo mohou pomocí aplikačního rozhraní SAGE2 (viz. 5.3.2) ovládat prostředí vzdáleně.

V současné době je volně k dispozici velké množství websocketových klientů. Značná část z nich je vázána na nestandardní knihovny (např.: boost) a jsou velice komplexní a složité. Takové knihovny nejsou vhodné vzhledem k povaze praktické části, proto byla vybrána odlehčená knihovna easywsclient[24], která je založena na standardních C a C++ knihovnách. Řešení umí navázat spojení, odesílat a přijímat zprávy.

Kapitola 5

Možnosti a vybrané řešení

SAGE2 prostředí je implementováno jako websocketový server, na který je možno se připojit z více klientů. Každý klient může přes ustálené vzdálené spojení se serverem ovládat prostředí. K ovládání prostředí je možné využít javascriptové aplikace spustitelné v internetovém prohlížeči, nebo je možné využít SAGE2 API a ovládat tak prostředí pomocí zpráv odesílaných na server. Celkově tedy vyvstávají dvě možnosti jak implementovat problém bezdotykového ovládání prostředí:

- Možnost využívající linuxové prostředí X11[25]
- Možnost využívající SAGE2 API

Základem obou řešení je mapování pohybů a pohybových gest rukou na události kurzoru. Hlavní rozdíly v řešení jsou pak ve zpracovávání a popřípadě odesílání takových událostí.

5.1 X11

K ovládání kurzoru prostředí SAGE je možno využívat pohybů kurzoru v operačním systému. Javascriptová aplikace SageUI, která je spustitelná z internetového prohlížeče, funkci zachytávání událostí kurzoru operačního systému a mapování na události kurzoru prostředí SAGE2 podporuje.

Jedním z možných řešení je tedy vytvořit program, který umožní bezdotykového ovládání linuxového prostředí. Události bezdotykově ovládaného kurzoru budou poté zachytávány zmíněnou javascriptovou aplikací a tím bude umožněna interakce s prostředím SAGE2. K implementaci řešení lze využít knihovnu xdotool[26], která umožní posílat falešné události kurzoru, tzn. takové události, které nevytvořila počítačová myš, ale jakýkoli jiný vstup (v případě bezdotykové aplikace vstupem budou gesta a pohyby rukou). V tomto řešení nelze definovat speciální gesta pro prostředí SAGE2, protože veškerá gesta jsou vázána na události počítačové myši.

Takto implementované řešení je plně využitelné i se starší verzí vizualizačního prostředí SAGE1, protože není vázáno na aplikační rozhraní prostředí, ale na linuxové rozhraní X11.

5.2 SAGE2 API

SAGE2 přijímá klienty, kteří mohou ovládat toto prostředí. Rozdíl oproti předchozímu řešení je především v implementaci vlastního websocket klientu, který bude odesílat události kurzoru na server přímo, tudíž máme plnou kontrolu nad všemi zprávami, které se na server posílají. Také při navázání spojení nového klienta je možno server zprávou nakonfigurovat tak, aby posílal důležitá data pro správné fungování aplikace (rozlišení telestěny, apod.). Bohužel tyto zprávy se mohou s nově příchozí verzí SAGE2 změnit (především kvůli faktu, že je prostředí stále ve verzi beta) a poté by se musela změnit i implementace aplikace. Pro správný běh aplikace není nutno využívat linuxové distribuce s grafickým prostředím, řešení tedy může být méně náročné na výkon počítače než v předchozím případě.

Tato implementace je specificky využitelná pouze pro prostředí SAGE2, tudíž nepodporuje starší verzi zmíněného vizualizačního prostředí. Výhodou ovšem jsou více specifická gesta, která je možno implementovat a mapovat na události v prostředí.

5.3 Zvolené řešení

Řešení, které je použito v praktické části, vychází z implementace pomocí SAGE2 API. Hlavním důvodem je nezávislost na linuxovém grafickém prostředí, také plná kontrola nad odeslanými zprávami a větší volnost při vytváření gest. Nevýhodou je především nekompatibilita se starším prostředím SAGE1. Vyřešení tohoto problému by obnášelo implementaci TCP klienta, který bude odesílat správně formátované zprávy na SAGE1 server (náhrada websocket klienta). Lepší porovnání obou řešení je vidět v tabulce 5.1.

5.3.1 Vybrané zařízení

Pro správné fungování aplikace je nutné vybrat vhodné zařízení s hloubkovým senzorem. V sekci 3.1 jsou nejznáměnší dostupná zařízení se zmíněným senzorem popsána. Při implementaci praktické části bude použito zařízení Microsoft Kinect Xbox, které je v laboratoři SAGElab dostupné. Velké množství knihoven a ovladačů na operačních systémech Linux dává tomuto zařízení široké možnosti využití. Nevýhodou zařízení Kinect ovšem zůstává malé rozlišení hloubkových snímků. K dispozici také bylo zařízení Microsoft Kinect for Windows. Bohužel tato jiná verze Kinectu není příliš stabilní s dostupnými ovladači pro Linux.

Tabulka 5.1: Porovnání možností řešení

	X11	SAGE2 API
Výhody	Řešení je možné použít i pro původní verzi SAGE1 bez zásahu do implementace	Není nutné vlastit linux s GUI, Programátor může přesně definovat zprávy odesílané na SAGE2, gesta mohou být definována přesněji vzhledem k aplikaci
Nevýhody	Závislost na linuxovém grafickém prostředí, odesílání zpráv je černá skříňka, není možné definovat přesná gesta pro prostředí	Nefunguje s původní verzí SAGE1, příchod nové verze SAGE2 může způsobit problémy

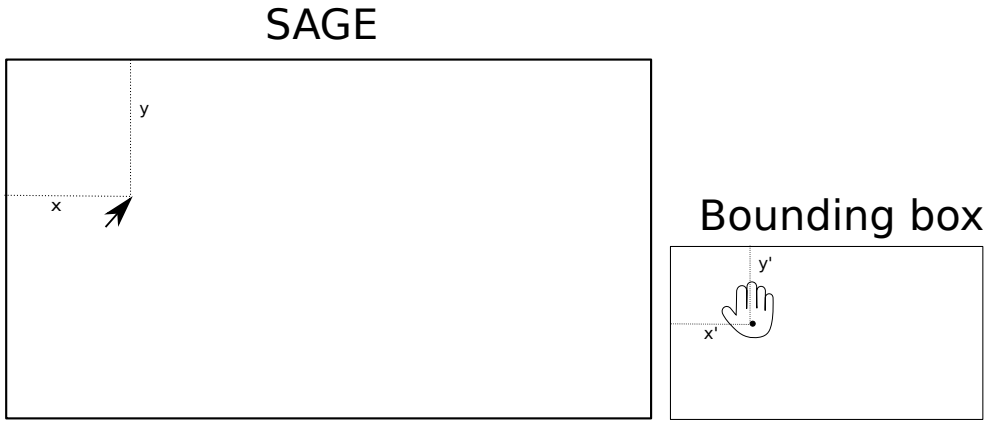
5.3.2 Aplikační rozhraní

Aplikační rozhraní poskytuje základní příkazy pro správu kurzoru v prostředí SAGE2 (posun kurzoru, simulace kliknutí, simulace kolečka, zapnutí a vypnutí kurzoru). API prostředí SAGE2 není prozatím zdokumentované, proto je nutné vyjít ze zdrojových kódů prostředí, které jsou volně k dispozici[4]. Důležité příkazy pro ovládání prostředí SAGE2 pomocí JSON zpráv (viz. 4.1) jsou:

- **addClient** Přidání a nastavení nového klienta na serveru
- **startSagePointer** Zapnutí kurzoru v prostředí
- **ptm** Posun kurzoru o určitý počet pixelů
- **pointerPress** Událost zmáčknutí tlačítka myši
- **pointerRelease** Událost puštění zmáčknutého tlačítka myši
- **pointerScrollStart** Zapnutí stavu hýbání s kolečkem
- **pointerScroll** Událost hýbání kolečkem o určitý počet řádek
- **stopSagePointer** Vypnutí kurzoru v prostředí

5.3.3 Přemapování pohybů a gest rukou na události kurzoru prostředí

Zařízení Kinect 4.5.1 při použití knihovny NiTE2 navrací pozice rukou ve světových souřadnicích definovaných zařízením. Hlavním problémem nastává,



Obrázek 5.1: Bounding box ruky

jak přemapovat tyto světové souřadnice a pohyb v nich na pozici a pohyby kurzoru v prostředí SAGE2. Řešení které je v praktické části zvoleno omezuje pohyb rukou pomocí definovaného bounding boxu[7]. Bounding box se musí definovat při detekci ruky, která bude mapovaná na pohyb kurzoru a to podle pozice kurzoru na obrazovce. Zároveň má box stejný poměr stran jako rozlišení telestěny, protože při zachování poměru bude pohyb kurzoru v prostředí plynulý do všech stran (tzn. stejně rychlý při jakémkoli směru pohybu). Na obrázku 5.1 je vidět vytvoření bounding boxu okolo ruky. Je zde také zřejmé, že box se vytvořil tak, aby ruka byla poměrově ve stejné pozici v bounding boxu jako kurzor v prostředí SAGE2. Šířka boxu bude zadávána z konfiguračního souboru. Při pohybu rukou se spočítá relativní pozice rukou v bounding boxu, přepočítá se na pixely ve vhodném poměru a odešle se na server.

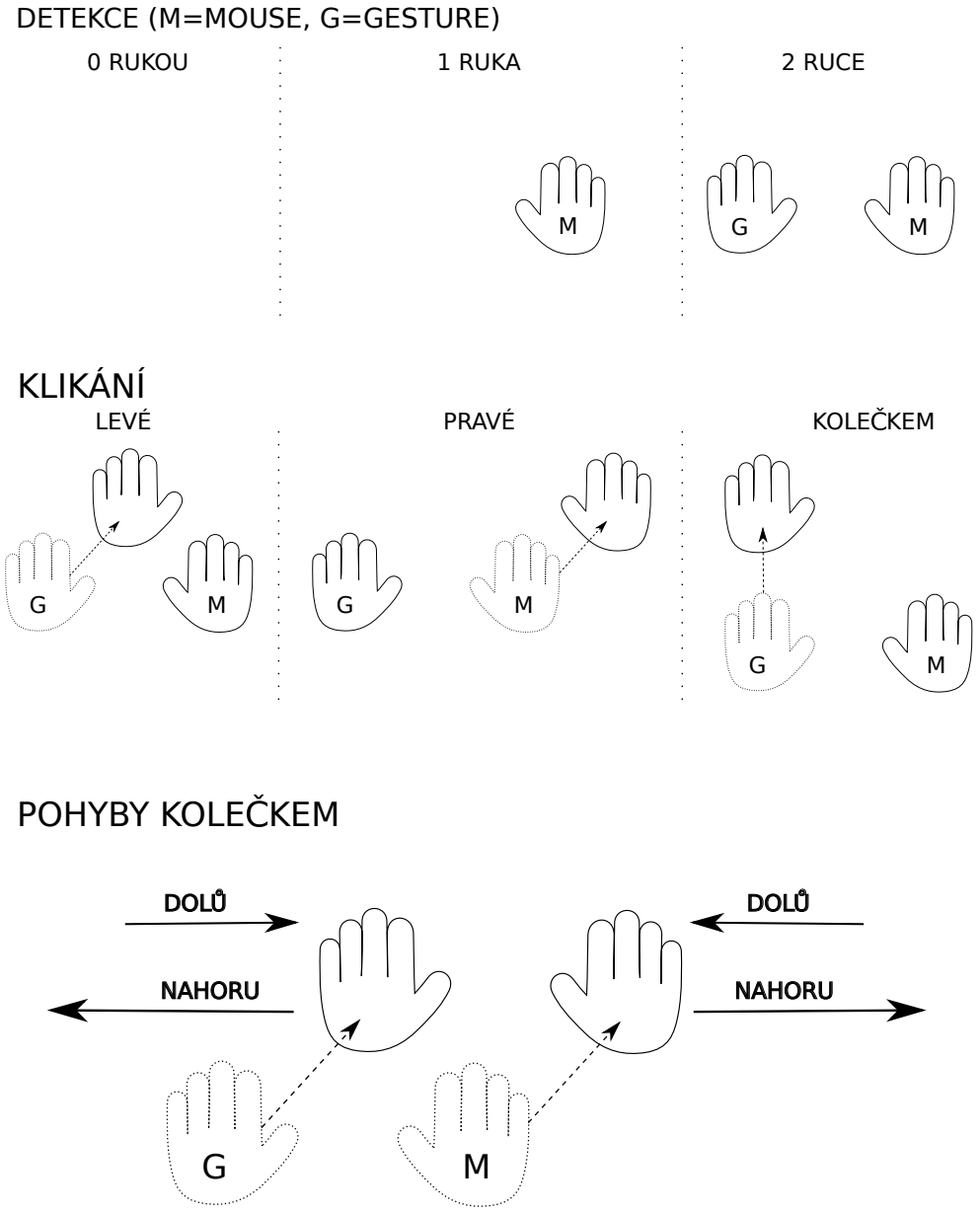
Gesta rukou budou zachycena programem. Pokud program zachytí gesto, odešle vhodnout zprávu na SAGE2 server.

5.3.4 Gesta

Jádro aplikace pro bezdotykové ovládání je postaveno na správné detekci gest rukou. Nadefinovaná gesta by měla být co nejvíce intuitivní tak, aby uživatel při práci s aplikací nemusel o prováděných gestech příliš přemýšlet a mohl se soustředit na práci s prostředím. Při návrhu gest bylo vzpomenuto především na gesta z jiných prostředí (některá gesta jsou inspirována dotykovými prostředími). Podle funkčích požadavků (viz. 3.2.2) jsou přesně vymezená gesta, která musí být implementována pro plnohodnotné využití prostředí SAGE2.

Při návrhu gest je také zřejmé, že pro některá gesta bude nutné mít zaměřené obě ruce. Při trackování obou rukou je poté nutné určit, na kterou ruku bude mapován pohyb kurzoru (mouse hand) a která ruka bude pouze podpo-

[7]Neviditelné hranice ve světových souřadnicích kolem rukou.



Obrázek 5.2: Definovaná gesta

rovat prováděná gesta (gesture hand). Definovaná gesta jsou nejlépe popsaná na obrázku 5.2.

5.3.4.1 Detekce

Při detekci program musí určit, na kterou ruku má být mapován pohyb kurzoru a která bude pouze podporovat gesta. Nejlepším řešením je první naleze-

nou a detekovanou ruku určit jako „mouse hand“ a teprve druhou nalezenou ruku jako „gesture hand“. Z řešení vyplývá, že uživatel bude moci při používání aplikace sám určit, na kterou ruku bude mapován pohyb kurzoru.

5.3.4.2 Klikání

Nejvíce intuitivní gesto pro klikání je přesun ruky vpřed, jako by uživatel chtěl sáhnout na prvek, na který chce kliknout. Simulace klikání levého a pravého tlačítka bude mapována na pohyby dopředu levé a pravé ruky uživatele. Klikání kolečkem bude mapováno na pohyb ruky „gesture hand“ nahoru.

5.3.4.3 Scrollování

Scrollování[8] je důležitou součástí ovládání prostředí SAGE2. Z mobilních dotykových zařízení je uživatel nejčastěji zvyklý na roztahování a stahování prstů při zmenšování a zvětšování prvků v prostředí. Aby bylo ovládání co nejvíce intuitivní, i v této aplikaci se změnšování a zvětšování prvků v prostředí bude řešit podobným gestem a to oddalováním a přibližováním rukou. Jelikož jsou obě ruce sledovány, nemuselo by být zaměření gesta pro scrollování přesné, proto zde bude vytvořeno gesto, pomocí kterého aplikaci pro bezdotykové ovládání přepneme z módu „hand tracking“ do módu „scrolling“.

[8] Pohyb kolečka nahoru a dolů

Kapitola 6

Implementace

Samotná implementace aplikace se co nejpřesněji drží výsledků přechozích kapitol analýzy 3, návrhu 4 a vybraného řešení 5.3. Na rozdíl od předchozích kapitol již nepopisuje, jak by se měl daný problém řešit, ale vysvětluje konkrétní implementaci prototypu aplikace.

Z návrhu vychází, že je vhodné aplikaci rozdělit na několik menších problémů a ty poté řešit zvlášť. Rozdělením získáváme větší možnosti při testování aplikace, kdy je možné otestovat dopodrobna jednotlivé části před tím, než se spojí v celek. Po testovací fázi se částečná řešení spojí a vznikne tak aplikace pro bezdotykové ovládání prostředí SAGE2. Kapitola popisuje nejen implementaci jednotlivých částí, ale i celkového produktu bakalářské práce. Popis je zaměřen na jednotlivé třídy, interakce mezi třídami a funkce vytvořené při programování aplikace. Zároveň jsou v kapitole vysvětlené veškeré konstanty, direktivy preprocesoru[9] a další důležité části programu.

Zpočátku kapitoly je popsán dodržovaný styl a notace zdrojových kódů. Poté jsou dopodrobna rozebrány jednotlivé části, na které je aplikace rozdělena dle návrhu a konec kapitoly je zaměřen na spojení implementovaných částí v jeden funkční celek.

6.1 Styl zdrojových kódů

Při implementaci byla snaha dodržovat jednotný styl a notaci programování. Jednotlivé příkazy jsou logicky odsazeny a vytváří se tak srozumitelná struktura zdrojového kódu. V implementaci je projevena snaha, aby každá funkce měla pouze jednu konkrétní činnost, o kterou se stará (dle názvu funkce). Celkově se implementace zaměřuje na čistotu kódu, přehlednost a srozumitelnost.

Všechny názvy (tříd, funkcí, proměnných,...) až na výjimky užívají klasickou *CamelCase*[10] notaci a jsou samopopisné. Ve zdrojových kódech jsou

[9] Direktivy jsou předzpracovány před vlastní kompilací kódu.

[10] Slovní spojení je bez mezer, slova se oddělují velkým písmenem, např.: camelCase

využity speciální notace pro zvýšení přehlednosti a srozumitelnosti:

- **CClassName** Třída s názvem *ClassName*. Všechny názvy tříd jsou pro větší přehlednost definovány s prefixovým písmenem „C“. Po prefixu je použita klasická *CamelCase* notace.
- **TStructName** Struktura s názvem *StructName*. Názvy struktur jsou pro odlišení od tříd prefixovány písmenem „T“. Zbytek názvu vychází z klasické notace použité v celém zdrojovém kódu.
- **m_MemberVariable** Členská proměnná třídy/struktury s názvem *MemberVariable*. Všechny členské proměnné jsou definovány s prefixem „m_“. Tím se docílí odlišení od klasických lokálních proměnných použitých v kódu.
- **MESSAGE_PRINTED_MESSAGE** Aplikace komunikuje s uživatelem pomocí zpráv vypisovaných v terminálu. Zprávy jsou zadefinovany jako direktivy preprocesoru, tudíž názvy proměnných reprezentující zprávy jsou psané velkým písmenem. Pro přesnost mají všechny zprávy prefix „MESSAGE_“.
- **Enum hodnoty** Datový typ enum je v aplikaci hojně využíván. Pomocí něj se může měnit stav aplikace nebo typ sledované ruky. Všechny hodnoty enum jsou psané velkým písmenem. Podle enum typu poté obsahují názvy prefixy či postfixy. Příkladem takových enum hodnot může být SCROLLING_STATE, kde je definovaný postfix „_STATE“ či HAND_MOUSE s prefixem „HAND_“.

Zdrojové kódy jsou zdokumentovány pomocí doxygen komentářů[27]. Komentáře jsou obsaženy v hlavičkových souborech tříd a okomentovány jsou veškeré důležité části a prvky kódu. Dokumentace je generovatelná ze zdrojových kódů za použití aplikace doxygen[28]. Při použití doxygen se vytvoří složka s dokumentací ve formátu HTML.

6.2 Načtení konfiguračního souboru

Konfigurační soubor nastavuje základní hodnoty aplikace, které mohou být změněny uživatelem. Je zaměřen na nastavení síťového spojení, nastavení výstupní obrazovky a na hodnoty ovlivňující nastavení hand trackeru [11].

Při implementaci načítače konfiguračního souboru bylo nutné myslet na strukturu souboru ve formátu INI (viz. 4.2). Pro každou hodnotu ze souboru je nutné uložit atribut, ke kterému je hodnota přiřazena a také sekci, do které patří. Nejvýhodnější implementací je použít strukturu multimap ze standardních knihoven C++. Klíčem multimapy bude vždy konkrétní sekce ukládané

[11] Algoritmus pro sledování pohybu rukou

hodnoty. Druhým argumentem multimapy je poté struktura obsahující atribut a hodnotu k němu přiřazenou.

CCommonData.h Soubor obsahuje definovanou třídu pro načítání konfiguračního souboru. Veřejné rozhraní obsahuje pouze konstruktor, funkci pro načtení dat ze souboru do multimapy uložené ve třídě a funkce pro získávání těchto hodnot (jsou definovány dvě funkce v závislosti na získávaném datovém typu). V konstruktoru třídy jsou nastaveny veškeré základní hodnoty pro případ, že by byl konfigurační soubor poškozen nebo by nemohl být načten.

- *loadData()* je metoda, která načte hodnoty z konfiguračního souboru. Jako argument požaduje cestu ke konfiguračnímu souboru.
- *getStringValue()* navrací nalezené hodnoty atributu z dané sekce v řetězcovém formátu.
- *getNumberValue()* je alternativou předchozí funkce s rozdílem, že navrací hodnoty v číselném formátu.

Privátní rozhraní třídy obsahuje především funkce pro parsování řádků v konfiguračním souboru. Při načítání souboru je také nutné ověřit, zda jsou vstupy validní. Za validní vstup se považují jakékoli přípustné hodnoty k danému atributu. Pokud hodnoty v konfiguračním souboru nejsou validní, poté je stav oznámen uživateli a nevalidní vstupy jsou nahrazeny základními hodnotami. Z toho vyplývá, že aplikace při načtení nevalidních či poškozených hodnot ze souboru nepřeruší svou činnost, ale pokračuje dál se základními hodnotami. Bílé znaky nemají na validitu vstupu vliv. Validními vstupy (řádky) aplikace rozumí:

- **Komentáře** Řádky považované za komentáře zařínají znakem „;“. Metoda *isComment()* kontroluje, zda je řádek komentářem.
- **Hlavičky sekcí** Za hlavičku sekce je považován řádek, který je ve tvaru „[Section_Header]“. Pro kontrolu se používá metoda *isHeader()*.
- **Atributy a hodnoty** Hodnoty jsou vždy přiřazeny k nějakému atributu. Validní řádky s hodnotami mají tvar „atribut=hodnota“. Hodnota může být buď číslo, nebo řetězec. V případě řetězce je poté důležité, aby byl jednoslovný. Metoda *parseLine()* parsuje řádek s hodnotami a kontroluje validitu řádků a metoda *checkValueLegal()* kontroluje validitu jednotlivých hodnot.

6.3 Základní inicializace

Při každém novém spuštění aplikace je nutné inicializovat jednotlivé části aplikace. Inicializace je velmi důležitá část programu a je prováděna hlavní

třídou *CContactlessApp*. Bez této části aplikace by nebyla možnost navázat spojení se serverem či používat vybraných knihoven (znamenalo by to také nemožnost použití připojeného zařízení).

CContactlessApp.h Hlavní třída aplikace pro bezdotykové ovládání. Do veřejných metod patří konstruktor starající se o základní nastavení členských proměnných a destruktor, který provádí smazání alokovaných prostředků v případě, že uživatel sám nezavolá funkci pro ukončení aplikace. Hlavní veřejné rozhraní třídy bez konstruktoru a destruktoru je definováno:

- *initializeApp()* tvoří zapouzdření pro veškeré inicializace prováděné v aplikaci. Jako argument požaduje cestu ke konfiguračnímu souboru, kterou předá příslušné třídě.
- *mainLoop()* je metoda s hlavním cyklem aplikace. V každém průchodu cyklu se zpracovává hloubkový snímek získaný z připojeného zařízení. Z výsledků zpracování snímku se poté správně namapují pohyby rukou na události kurzoru prostředí SAGE2.
- *finalizeApp()* maže prostředky alokované hlavní třídou.

Privátní rozhraní třídy obsahuje funkce pro inicializaci jednotlivých částí programu. Veřejná metoda *initializeApp()* tvoří zapouzdření dílčích inicializací. Dílčí metody nastavení aplikace provádějí:

- Vytvoření třídy načítače souboru a načtení konfiguračního souboru. Proces provádí metoda *initializeConfig()*, která jako argument požaduje cestu ke konfiguračnímu souboru.
- Vytvoření třídy websocket klienta a připojení na SAGE2 websocket server zavoláním metody *initializeNetwork()*, která požaduje adresu a port cílového serveru.
- Načtení knihovny OpenNI2 (funkce pro inicializaci API této knihovny) a otevření připojeného zařízení, které bude snímat pohyby uživatele. Zpracovává metoda *initializeDevice()*.
- Podobně jako v předchozím bodě načtení NiTE2 a základní nastavení algoritmů pro extrakci pozic rukou z hloubkových snímků metodou *initializeHandTracker()*.

6.4 Mapování gest na události kurzoru prostředí

Při detekování jednotlivých gest uživatele aplikace zjistí typ provedeného gesta a následně odešle SAGE2 serveru příslušnou zprávu s událostí, kterou dané gesto představuje. Události jsou posílané ve formátu JSON zpráv (viz. 4.1)

přes udržované spojení s websocket serverem. O proces detekce a přemapování gest se starají třídy *CHand*, *CMouseMapper* a *CTracker*.

CHand.h Třída reprezentující sledovanou ruku a uchovávající informace o pozici ruky načtené z připojeného zařízení a zároveň také identifikátory ruky jako jsou ID a typ ruky. Pokud je typ ruky „hand mouse“ poté také definuje hranice pohybu ruky (viz. 5.3.3). Veřejné rozhraní této třídy poskytuje klasické metody pro nastavení a získání ID, typu a pozice ruky. V konstruktoru je požadováno zadání vzdálenosti, kterou se musí ruka urazit, aby bylo detekované gesto klikání. Dále rozhraní poskytuje:

- *setBoundingBox()* metoda nastavuje hranice pohybu ruky typu „hand mouse“. Ke správnému vytvoření hranic požaduje zadat do argumentů šířku hranic, poměr stran a relativní pozici kurzoru v prostředí SAGE2.
- *getRelativePosition()* navrátí procentuální pozici ruky v definovaném bounding boxu.
- *checkForClickGesture()* metoda kontroluje, zda nebylo provedeno klikací gesto. Kontrola probíhá na základě inicializační pozice uložené při prvním načtení ruky a vzdálenosti zadané v konstruktoru. Argumentem funkce je typ klikacího gesta.

Privátní rozhraní třídy obsahuje pouze uchovávané členské proměnné. Žádné speciální privátní metody nebylo nutné definovat.

TBoundingBox.h Struktura, která reprezentuje hranice pohybu ruky ve světových souřadnicích zařízení. Podrobný popis rozhraní struktury:

- *initializeBox()* nastaví nové hranice okolo ruky. Hranice se vytvoří dle zadaných argumentů levého horního bodu hranice, šířky hranice a výšky hranice.
- *checkHandPosition()* dle zadané pozice ruky určí, zda se ruka nachází v bounding boxu. Pokud se ruka nachází mimo box, poté je pozice ruky upravena na hraniční souřadnice boxu.
- *getRelativePositionInBox()* navrací dle zadané pozice ruky relativní pozici ruky v bounding boxu.

CMouseMapper.h Reprezentace kurzoru v prostředí SAGE2, která zároveň uchovává i informace o výstupní obrazovce. Třída umožňuje odesílat na server veškeré události myši, které prostředí umí zpracovat. Při práci s pozicí kurzoru je nutné dbát na souřadnicový systém obrazovky (obrázek 6.1), který se odlišuje od systému zařízení v orientaci os. Z důvodu odesílání událostí uchovává třída ukazatel na websocket klient, přes který se odesílají zprávy na server. Krom konstruktoru a destruktoru, obsahuje třída rozhraní:

- *initialize()* metoda z třídy, která uchovává data načtená z konfiguračního souboru, načte informace důležité pro kurzor prostředí SAGE2 a uloží je do členských proměnných třídy. Argumenty metody jsou ukazatele na třídu websocket klienta a na třídu, která načítá data z konfiguračního souboru.

- *startSagePointer()* a *stopSagePointer()* jsou metody, které pošlou serveru zprávu s informací o zapnutí/vypnutí kurzoru prostředí.

- *startScrolling()* posílá serveru zprávu o začátku pohybu kolečkem myši.

- *getCurrentMousePositionRelative()* navrátí pozici kurzoru v prostředí relativně vzhledem k rozlišení obrazovky.

- *getScreenAspectRatio()* navrátí poměr rozlišení obrazovky. Poměrem se rozumí podíl šířky ku výšce rozlišení obrazovky.

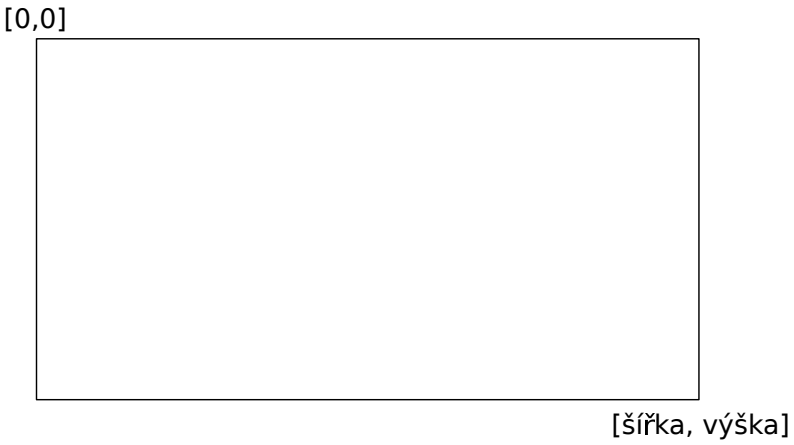
- *setMouseRelative()* dle zadané relativní 2D pozice ruky vypočítá novou pozici kurzoru v pixelech a uloží ji do členských proměnných. Zároveň vypočte rozdíl mezi novými a posledními známými souřadnicemi a odešle jej na server ve zprávě informující o pohybu kurzoru.

- *setMouseDown()* a *setMouseUp()* metody odesílají události zmačknutní a uvolnění tlačítka myši dle typu tlačítka zdaného v argumentu metody. Třída uchovává stavy zmáčknutých tlačítek, tudíž se události na server odesílají pouze jednou (do opětovného uvolnění/zmáčknutí tlačítka).

- *setScrollingDelta()* odešle na server informaci o rozdílu pohybu kolečka.

- *isPressed()* navrátí stav tlačítka (tzn. zda-li je tlačítko zmáčknuté) zadaného v argumentu metody.

CTracker.h Hlavní třída procesu přemapování pohybů rukou na události kurzoru prostředí SAGE2. Třída zpracovává hloubkové snímky načtené ze zařízení a zpracovává je. Zpracováním se myslí především extrahování pozic rukou ze snímků. Zároveň obsahuje informace o stavu, ve kterém se nachází (stavy mohou být „mouse tracking“ a „scrolling“). Dle stavu se poté na extrahovaná data ze snímků volají příslušné metody. Třída uchovává také informace potřebné ke správnému průběhu algoritmu sledování pozic rukou. Dále ukládá aktuální načtený snímek, seznam sledovaných rukou (třídy *CHand*) či třídu *CMouseMapper* starající se o reprezentaci kurzoru v prostředí SAGE2. Uložený seznam sledovaných rukou může obsahovat maximálně dvě ruce. Veřejné rozhraní třídy se stará především o inicializaci třídy, zapnutí a vypnutí hand trackeru a načtení snímků ze zařízení a jejich zpracování.



Obrázek 6.1: Souřadnice obrazovky

- *setDevice()* nastaví ukazatel na třídu reprezentující zařízení (knihovna OpenNI2), ze kterého se budou načítat hloubkové snímky vhodné ke zpracování.
- *initializeHandMapping()* uloží do členských proměnných data načtená z konfiguračního souboru. Daty jsou například sensitivita zařízení, vzdálenost potřebná k detekování gesta klikání či faktor vyhlazení pohybu rukou načtených ze zařízení. Zároveň inicializuje třídu *CMouseMapper*. Argumenty jsou ukazatele na websocket klienta a třídu načítající data z konfiguračního souboru.
- *start()* a *stop()* metody pro zapnutí a vypnutí algoritmu hand trackeru.
- *readFrame()* přečte snímek za zadaného zařízení, zkontroluje zda je snímek validní a uloží jej do členských proměnných třídy. V případě nevalidity se snímek přeskakuje a načítá se další, uživatel je o přeskočení snímku informován příslušnou zprávou.
- *releaseFrame()* uvolní prostředky alokované načteným snímkem. Metoda se volá na konci každého cyklu.
- *handDetection()* detekuje nově načtenou ruku. Pokud je ruka nově detekována a zároveň se uživatel nachází v dostatečné vzdálenosti od zařízení snímající pohyb (viz. 4.5.2), poté se ruka zařadí mezi sledované ruce.
- *handMapping()* patří mezi hlavní metody celé aplikace. Úkolem metody je extrahovat pozice všech rukou z načteného snímku. Po extrakci se ruce musí zpracovat na základě faktu zda je ruka nově načtená, ztracená nebo sledovaná a také na základě stavu třídy *CTracker*. Zpracování jednotlivých stavů probíhá privátními metodami třídy.

- *closeTracker()* uvolní všechny prostředky alokované třídou. Volá se na úplném konci programu.

Privátní metody obsahují pomocné funkce při implementaci. Do těchto metod se také řadí metody pro zpracování rukou načtených ze snímků v závilosti na stavu, v jakém se ruka a třída nacházejí (tzn. metody mohou zpracovávat nově načtené ruce, ztracené ruce, apod.). Privátní rozhraní třídy obsahuje funkce:

- *inTrackingRange()* na základě zadané pozice ruky určí, zda se uživatel nachází ve správné vzdálenosti od zařízení. V argumentu požaduje pozici ruky jejíž validita se kontroluje.
- *getHandDistance()* navrátí euklidovskou vzdálenost[29] pozic sledovaných rukou (pozice jsou vektory vycházející z počátku souřadného systému). Vzdálenost se používá při provádění gesta hýbání kolečkem.
- *updateTrackerState()* kontroluje a případně aktualizuje stav třídy. Stav třídy je možno přepínat pomocí definovaného gesta. Pokud dojde k přepnutí stavu do „scrolling“ je nutné přestat mapovat pohyb rukou na pohyby kurzoru a začít sledovat vzdálenost mezi rukami reprezentující hýbání kolečkem. V případě přepnutí do „mouse tracking“ se opět zapne mapování pohybů rukou na pohyb kurzoru.
- *processNewHand()* zpracovává nově sledovanou ruku. Ruka je nově sledovaná, pokud se nevyskytovala v žádném předchozím snímku. Při takto načtené ruce je nutné určit typ ruky (zda je ruka určená k mapování pohybů kurzoru či k provádění gest). Metoda tedy vytvoří novou instanci *CHand*, nastaví její potřebné atributy a přidá ji do seznamu sledovaných rukou. Argumentem metody je ukazatel na ruku načtenou ze snímku.
- *processHandsMovements()* podle typu aktuálně extrahované ruky ze snímku zpracuje pohyb této ruky. Metoda sleduje, zda nebylo rukou provedné příslušné gesto klikání a případně jej přemapuje na události *SAGE pointeru*. Jako argument přijímá ukazatel na ruku ze seznamu rukou, která je aktivní (právě extrahovaná).
- *processClickGesture()* kontroluje, zda nebylo provedeno gesto pro klikání a případně jej přemapuje na příslušnou událost. Ke správnému fungování metody je nutné předat ji jako argument aktivní ruku ze seznamu, typ klikacího gesta a tlačítko myši na které je událost mapována.
- *processScrolling()* zpracovává pohyby rukou, pokud je třída ve stavu *scrolling*. Sleduje se především vzdálenost rukou, která se mapuje na pohyby kolečka. Událost pohybu kolečka dolů je mapována na přibližování rukou, oddalování rukou je poté mapováno na opačný pohyb kolečkem myši.

- *getActiveHand()* prohlédne seznam rukou a porovná aktuálně extrahovanou ruku zadanou v argumentu s rukami uloženými v seznamu. Pokud se tam již nachází, vrátí se ukazatel na ruku ze seznamu, v opačném případě se vrací *NULL*.
- *updateHandBox()* aktualizuje bounding box (viz. 6.4) ruky, jejíž pohyb je mapován na pohyby kurzoru. Metoda se volá především při detekování nově sledované ruky nebo při přepnutí stavu do „mouse tracking“.
- *freeHandsList()* uvolní prostředky alokované rukou, jejíž typ je zadan v argumentu metody.
- *freeLostedHand()* v případě ztráty ruky (například pokud je ruka mimo sledovaný rozsah zařízení) se uvolňují prostředky ztracené ruky a zároveň se taková ruka musí odstranit ze seznamu sledovaných rukou. Argumentem funkce je ukazatel na ruku ze seznamu, která se ztratila.

6.5 Websocket klient

Každé pohybové gesto je mapováno na události *SAGE pointeru*, které jsou odesílány na server. K odesílání je nutná implementace websocket klienta, jehož úkolem je navázat spojení se serverem SAGE2 dle zadané adresy a portu a odesílat jednotlivé zprávy, které musí být v JSON formátu (viz. 4.1). V případě aplikace pro bezdotykové ovládání není nutností implementovat metody, které budou zprávy přijímat.

CSageNetClient.h Třída reprezentující websocket klienta, která uchovává URL adresu a ukazatel na konkrétní websocket reprezentující spojení se serverem. V konstruktoru očekává třída zadané hodnoty adresy a portu, ze kterých vytvoří URL, na kterou se klient připojí. Další veřejné rozhraní se stará především o navázání spojení, inicializaci klienta na serveru a odesílání zpráv server.

- *connectToServer()* je metoda, která vytvoří websocket reprezentující spojení mezi klientem a serverem, po kterém se budou odesílat zprávy.
- *initialize()* zaregistruje nového klienta na serveru SAGE2. Každý klient před komunikací se serverem musí poslat zprávu, ve které informuje server o hodnotách, které bude odesílat a které chce přijímat, tzn. klient se na serveru inicializuje.
- *closeNetwork()* uzavře spojení se serverem a zároveň uvolní veškeré prostředy alokované klientem.
- *sendMessage()* vytváří zprávy ve formátu JSON, které jsou serveru odesílány dle zadaných argumentů. Metoda je přetížená, tudíž přijímá dva

druhy argmentů. Prvním argumentem je vždy název příkazu ze SAGE2 API 5.3.2. Dalšími argumenty mohou být až 2 řetězce s daty, nebo 1-2 číselné hodnoty. Dle počtu argumentů a jejich datových typů je vytvořena příslušná zpráva, která je odeslána na server.

Privátní rozhraní třídy obsahuje pouze jednu *sendMsg()* metodu, která odesílá vytvořené zprávy z privátních metod na server přes vytvořené websocketové spojení.

6.6 Souhrn implementace

Sekce popisuje pomocné třídy a funkce, které byly použity zejména kvůli zjednodušení implementace a dodržení čistoty kódu a také se zabývá spojením jednotlivých popsaných částí do jednoho funkčního celku. V předchozích popisech jednotlivých částí již byly některé návaznosti zřejmé (předané ukazatele, apod.), avšak spojení nebyla popsána podrobněji.

6.6.1 Pomocné třídy a funkce

CException.h Pomocná třída reprezentující výjimky. Třída uchovává zprávu výjimky, která je nastavena v konstruktoru třídy. Na rozdíl od standardních výjimek třída upravuje zprávu, která byla nastavena při vytváření instance třídy. Obsahuje pouze jednu funkci *getException()* získávající ze třídy výjimku v řetězcovém formátu. Uživateli je poté zachycená výjimka zobrazena ve formátu:

```
(ERROR) Zpráva výjimky
```

Common.h Soubor obsahující společné funkce, proměnné typu enum a direktivy preprocesoru pro celou implementaci aplikace. Tento soubor je připojen do více hlavičkových souborů, které využívají tyto společné části kódu.

Obsahuje také direktivy preprocesoru, ve kterých jsou uložené zprávy zobrazené uživateli (zprávy skrze které aplikace komunikuje s uživatelem). Zprávy mohou být informační, upozorňující či vyhozené výjimkou.

- „Informační zprávy“ nejčastěji oznamují uživateli stav aplikace. Jsou to například zprávy, které jsou vypsány při startu aplikace.
- „Upozorňující zprávy“ dávají uživateli na vědomí, že aplikace nemusí fungovat správně (např.: při nevalidních hodnotách v konfiguračním souboru jsou načtené základní hodnoty). Upozornění nejsou ovšem kritická a nepreruší běh aplikace.

```
(WARNING) Zpráva upozornění
```

- „Zprávy vyhozené výjimkou“ jsou zprávy, které se používají před přerušením běhu aplikace, aby uživatel věděl, proč aplikace nemůže dále pokračovat.

Soubor dále obsahuje definované enum proměnné. V aplikaci jsou využity zejména k označení stavu třídy *CTracker*, tlačítek myši, typů gest či typů sledované ruky. Posledními prvky obsaženými v souboru jsou společné funkce. Funkce jsou často použity ve více částech zdrojového kódu. Přesunutím jejich definic do společného souboru se zamezí duplicitám v kódu.

- *handleConsoleInput()* kontroluje, zda je aplikace řádně spuštěna. Argumentem funkce je počet načtených parametrů z příkazové řádky a pole obsahující tyto parametry. Při spouštění aplikace je nutné přidat jako parametr cestu ke konfiguračnímu souboru:

```
./nazev programu -f cesta/ke/konfiguracnimu/souboru
```

- *printToConsole()* je funkce vypisují na standardní výstup zprávy uživateli. Argumentem funkce je zpráva, která bude vypsána.
- *enumToStrButton()* převádí enum typ *buttons* reprezentující jednotlivá tlačítka myši do řetězcového formátu.
- *keyboardHit()* je neblokující funkce, která čeká na vhodné uhození klávesy. Funkce je použita pro ukončení hlavního cyklu aplikace. Pro ukončení aplikace jsou definovaná písmena „Q“ a „q“.
- *toNumber()* převádí zadaný vstup v řetězcovém formátu na číslo. Pokud řetězec nereprezentoval kladné číslo, poté je navrácena -1, v opačném případě je navrácena číselná hodnota.
- *isNumber()* zkoumá, zda zadaný řetězec reprezentuje číslo. Pomocí argumentu lze specifikovat, jestli řetězec má představovat celé číslo, nebo číslo s plovoucí desetinou čárkou.
- *toString()* převádí zadané číslo do řetězcového formátu.

6.6.2 Spojení implementovaných částí

Běh začíná funkcí *main*, která zkontroluje, zda byla aplikace spuštěna se správnými parametry a vytvoří instanci hlavní třídy *CContactlessApp*. Hlavní třída inicializuje všechny potřebné části aplikace a knihovny OpenNI2 a NiTE2 4.5.

Při inicializacích jsou vytvořeny a nastaveny instance tříd *CCommonData* (načtou se hodnoty z konfiguračního souboru a uloží se do členských proměnných třídy), *CTracker* (nastaví se zařízení a základní hodnoty hand tracking algoritmu) a *CSageNetClient* (vytvoří se spojení na server). Poté se spustí hlavní cyklus aplikace, který je ukončitelný pomocí vstupu, na který čeká funkce *keyboardHit()*. V cyklu je třídou *CTracker* přečten a zpracován hloubkový snímek načtený ze zařízení. Po načtení snímku se detekují nové ruce *CHand*, které se přidají do seznamu sledovaných rukou jejichž pozice bude nadále sledována. V seznamu mohou být nanejvýš dvě sledované ruce, kdy jedna ruka je typu „hand mouse“ a druhá typu „hand gesture“. *CTracker* poté aplikuje algoritmy pro přemapování pohybů těchto rukou uložených v seznamu na události *SAGE pointeru*. Třída reprezentující kurzor v prostředí SAGE2 je *CMouseMapper*, která zároveň odesílá detekované události na server skrze websocket klienta implementovaného třídou *CSageNetClient* a který je předán do mapperu pomocí ukazatele. Formát těchto odesílaných zpráv je JSON.

Uživatel je o stavu aplikace informován pomocí zpráv vypisovaných na standardní výstup funkcí *printToConsole()*. Při jakékoli závažné chybě je vyhozena výjimka *CException*. Zprávy pomocí kterých aplikace komunikuje s uživatelem jsou uloženy ve společném souboru *Common.h*.

Kapitola 7

Testování

Testování aplikace by mělo být součástí každého vývoje. Už při vývoji aplikace je nutné jednotlivé části aplikace otestovat (např.: pomocí náhodných dat, testy spolehlivosti spojení apod.). Důležité jsou ovšem také testy programu jako celku.

Kapitola se zabývá především testováním ovládání a zkoumá, zda je intuitivní, srozumitelné pro uživatele a především jestli je splněn cíl aplikace ovládát plnohodnotně prostředí SAGE2 bezdotykově. Zároveň se zaměřuje na konkrétní gesta a jejich zhodnocení uživateli.

7.1 Popis testování

K otestování aplikace byla vybrána skupina pěti bakalantů, kteří zaměřují své bakalářské práce na vizualizační prostředí SAGE2. Připravené testy měly ověřit především intuitivnost gest a ovládání všech prvků v prostředí. Studentům byly poskutnuté instrukce k bezdotykovému ovládání prostředí, poté každý z nich dostal stejně zadaný úkol, který zahrnoval použití všech gest detekovatelných aplikací. Ke splnění těchto těchto úkolů bylo nutné využít téměř všechny ovládací prvky prostředí.

Scénář testování V prostředí SAGE2 byla spuštěna aplikace *Google Maps* zobrazující mapu světa[30]. Úkoly, které bylo potřeba splnit:

- Otevřít prohlížeč obrázků s konkrétní fotografií z menu (gesto levého kliku pro zobrazení menu, gesto pravého kliku pro výběr)
- Přesunout aplikaci *Google Maps* (držení gesta levého kliku a hýbání rukou, na kterou je namapován pohyb kurzoru)
- Zvětšit aplikaci *Google maps* (gesto pro přepnutí do stavu „scrolling“ a poté oddálení rukou)

- Přepnout stav do interakce s aplikací (gesto pro změnu stavu do módu interakce)
- Oddálení mapových podkladů (přepnutí do stavu „scrolling“ a poté přiblížení rukou)

Po otestování byla studentům předložena anketa. Anketa obsahovala možnost obodování jednotlivých gest a zhodnocení intuitivnosti ovládání, dále se dotazovala na úspěšnost splněných úkolů a problémy, které při ovládání nastávaly. Vyplňování ankety bylo anonymní.

7.2 Výsledky testování

Jelikož anketa byla anonymní, odpovědi studentů byly pro přehlednost označeny čísly 1-5. Obodování jednotlivých gest[12] lze vidět v tabulce 7.1. Další odpovědi na otázky:

- *Je ovládání intuitivní?* Až na studenta 2 odpověděli všichni studenti, že ovládání je intuitivní.
- *Povedlo se Vám splnit zadané úkoly?* Všichni studenti kromě studenta 4 zvládli splnit zadané úkoly.
- *Měli jste problémy s ovládáním? Jaké?* Mezi problémy s ovládáním se především objevovaly odpovědi, které kritizovaly detekování jednotlivých gest, která se musí provádět v ose senzoru a ne v ose telestěny. Odpovědi také kritizovaly polohu rukou, které musí být stále zvednuté (student by preferoval gesta dlaní). Dalším problémem byla záměna rukou (změna ruky, na které je mapován pohyb kurzoru) v případě, že se ruce přiblížily maximálně k sobě. Poslední větší kritika studenta říkala, že by raději uvítal gesta na jednotlivé úkony v prostředí SAGE2, než mapování pohybů ruky na pohyby kurzoru.
- *Jaký je Váš celkový pocit z bezdotykového ovládání?* Studenti se shodli, že testování aplikace byla zajímavá zkušenost, která má jisté využití při prezentování či hraní her s použitím telestěny. Zároveň se také shodli, že bezdotykové ovládání má stále v mnoha ohledech rezervy.

7.3 Zhodnocení výsledků testování

Nejhůře ohodnoceným gestem bylo gesto pro zmenšení/zvětšení (neboli pro simulaci pohybů kolečkem). Hlavním důvodem bude problém, který byl často zmíněn v kritice a to že gesta musí být prováděna v ose senzoru a ne telestěny.

[12] Body mají hodnoty jako známky ve škole

Tabulka 7.1: Tabulka výsledků testování

	gesto zmenšení/zvětšení	gesto klikání	posouvání oken
student 1	3	1	1
student 2	4	2	3
student 3	2	2	2
student 4	2	2	1
student 5	4	4	4

Problém se s velkou pravděpodobností týká i hodnocení ostatních gest. Algoritmy detekce pohybových gest lze částečně vylepšit, avšak řešení problému je stále omezeno použitím zařízení pro snímání hloubkových snímků (zůstává problém s osou zařízení). Dále při problému záměny rukou při maximálním přiblížení selhává algoritmus extrakce pozic rukou z hloubkových snímků, řešením by mohlo být zařízení s vyšším rozlišením snímků. Dotazovaní by také uvítali gesta na jednotlivé úkony v prostředí než přemapování pohybu. Prostředí SAGE2 v současné implementaci není uzpůsobeno na ovládání pouhými gesty bez přemapování pohybů ruky na pohyby kurzoru (například by nebylo možné spouštět a vybírat aplikace, fotografie a videa). Zároveň SAGE2 API je prozatím uzpůsobeno na přijímání základních událostí kurzoru, tudíž je řešení v současné době neproveditelné bez nutnosti úpravy zdrojových kódů SAGE2.

Ve výsledku bylo ovládání intuitivní a téměř všem testovaným osobám se podařilo splnit zadané úkoly. Problémy zmíněné v kritice jsou až na vylepšení stávajících algoritmů prozatím těžko řešitelné vzhledem k zadání práce a zvolenému řešení, jelikož se často naráží na hranice možností zvoleného zařízení či knihoven použitých v implementaci.

7.3.1 Náměty na vylepšení detekce gest

Souhrn námětů popisuje navrhované řešení na největší problémy aplikace, které se objevily během testování uživateli.

- **Vylepšení algoritmů** Testovaní uživatelé měli často problém s detekováním jednotlivých gest, které neprováděli v ose senzoru. Ovšem problémem jsou také implementované algoritmy, které by bylo vhodné v dalším vývoji vylepšit a zvýšit tak přesnost detekování jednotlivých gest.
- **Zařízení** V citlivosti snímání se také naráží na problém malého rozlišení hloubkových snímků, které poskytuje zařízení Kinect. Nejčastěji se tento problém vyskytne při velkém přiblížení rukou. Prozatím neexistují správné ovladače pro pokročilejší verzi zařízení Kinect v2 s vyšším rozlišením, ovšem lze předpokládat, že se v dohledné době objeví. Poté by bylo vhodné využít Kinect v2 namísto staré verze.

Závěr

Cílem bakalářské práce bylo seznámení se s dosavadními řešeními bezdotykového ovládání vizualizačních prostředí a navrhnout nové vlastní řešení pro síťovou a multimediální laboratoř Českého vysokého učení technického (SAGElab), která využívá telestěny s prostředím Scalable Amplified Group Environment (SAGE2). Součástí práce také bylo implementování prototypu aplikace umožňující bezdotykové ovládání zmíněného prostředí, tzn. umožňující ovládání kurzoru prostředí a zároveň pomocí vhodných gest simulující veškeré události myši.

Záměr praktické části byl splněn, tudíž byl vytvořen funkční prototyp aplikace umožňující plnou bezdotykovou interakci s prostředím SAGE2. Následující uživatelské testování prototypu odhalilo nedostatky zejména při detekování jednotlivých pohybových gest, avšak i přes to se implementovaná gesta zdála být uživatelům intuitivní a celkový dojem z používání aplikace byl spíše pozitivní. Výsledky z testování poslouží jako podklady pro další vývoj aplikace, který bude zaměřen na vylepšení algoritmů pro detekci pohybových gest. Z důvodu rychle se rozvíjejících technologií lze předpokladat, že již v blízké době bude možné použít i kvalitnější hloubkové zařízení s větším rozlišením snímků. Zároveň jsou zdrojové kódy veřejně dostupné[31], tudíž se na vývoji aplikace může podílet více programátorů.

Literatura

[1] University of Illinois at Chicago: *Electronic visualization laboratory [online]*. 2015, [cit. 2015-03-24]. Dostupné z: `https://www.evl.uic.edu/`

[2] CESNET, FIT a FEL ČVUT: *Síťová a multimediální laboratoř [online]*. 2015, [cit. 2015-04-01]. Dostupné z: `http://sage.fit.cvut.cz/`

[3] Sage commons: *Scalable adaptive graphics environment (SAGE) [online]*. 2015, [cit. 2015-04-01]. Dostupné z: `http://sage.sagecommons.org/`

[4] Sage commons: *Scalable amplified group environment (SAGE2) [online]*. 2015, [cit. 2015-03-23]. Dostupné z: `http://sage2.sagecommons.org/`

[5] Norris, S.: *Depth frame image [online]*. Norris Labs, 2012, [cit. 2015-03-31]. Dostupné z: `http://www.norrislabs.com/`

[6] Kaazing Corporation: *Websocket [online]*. 2013, [cit. 2015-03-23]. Dostupné z: `https://www.websocket.org/`

[7] evltube: *Kinect for SAGE [online]*. YouTube, LLC, 2011, [cit. 2015-03-23]. Dostupné z: `https://www.youtube.com/watch?v=oFQeszkCaPU`

[8] UIC, E. V. L.: *Omicron SDK [library]*. GitHub, Inc, 2015, [cit. 2015-03-23]. Dostupné z: `https://github.com/uic-evl/omicron`

[9] Leap Motion, Inc: *Leap Motion [online]*. 2015, [cit. 2015-04-01]. Dostupné z: `https://www.leapmotion.com/`

[10] IK Multimedia US, LLC: *iRing - Motion controller for music apps and more [online]*. 2015, [cit. 2015-04-01]. Dostupné z: `http://www.ikmultimedia.com/products/iring/`

[11] Logbar, Inc: *Ring ZERO by Logbar [online]*. 2015, [cit. 2015-04-01]. Dostupné z: `http://www.logbar.jp/ring/en/`

[12] Nod, Inc: *Nod [online]*. 2015, [cit. 2015-04-01]. Dostupné z: https://www.nod.com/

[13] Microsoft: *Microsoft Xbox [online]*. 2015, [cit. 2015-03-23]. Dostupné z: http://www.xbox.com/cs-CZ/

[14] Microsoft: *Microsoft Kinect pro Xbox One [online]*. 2015, [cit. 2015-03-23]. Dostupné z: http://www.xbox.com/cs-CZ/xbox-one/accessories/kinect-for-xbox-one

[15] ASUSTeK Computer Inc.: *Asus Xtion Pro [online]*. 2015, [cit. 2015-03-23]. Dostupné z: http://www.asus.com/cz/Multimedia/Xtion_PRO/

[16] Microsoft: *Microsoft Kinect pro Xbox 360 [online]*. 2015, [cit. 2015-03-23]. Dostupné z: http://www.xbox.com/cs-CZ/Xbox360/Accessories/kinect

[17] Mlejnek, I. J.: *Analýza a sběr požadavků*. Fakulta informačních technologií, T9:105, Zimní semestr, 2014, [cit. 2015-03-23].

[18] Ecma International: *The JSON Data Interchange Format [online]*. Říjen 2013, [cit. 2015-03-23]. Dostupné z: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

[19] Microsoft: *Configure an Ini File Item [online]*. 2013, [cit. 2015-03-23]. Dostupné z: https://technet.microsoft.com/en-us/library/cc731332.aspx

[20] OpenKinect group: *OpenKinect project [library]*. 2012, [cit. 2015-03-23]. Dostupné z: http://openkinect.org/

[21] Occipital, Inc.: *OpenNI 2 Downloads and Documentation | The Structure Sensor [online]*. 2015, [cit. 2015-03-23]. Dostupné z: http://structure.io/openni

[22] PrimeSense: *Open source computer vision (OpenCV) [library]*. 2013, [cit. 2015-03-23]. Dostupné z: PrimeSense

[23] Itseez: *PrimeSense middleware library NiTE2 [library]*. 2015, [cit. 2015-03-23]. Dostupné z: http://www.openni.ru/files/nite/index.html

[24] Baird, D.: *Easy websocket client (easywsclient) [library]*. 2015, [cit. 2015-03-23]. Dostupné z: https://github.com/dhbaird/easywsclient

[25] X.Org Foundation: *X Window System (X11) [online]*. 2015, [cit. 2015-03-23]. Dostupné z: http://www.x.org/

[26] Sissel, J.: *Knihovna xdotool [library]*. 2015, [cit. 2015-03-23]. Dostupné z: https://github.com/jordansissel/xdotool

[27] Katedra počítačové grafiky a interakce, Fakulta elektrotechnická, České vysoké učení technické: *Jak psát komentáře v Doxygenu [online]*. 2014, [cit. 2015-03-23]. Dostupné z: `https://cent.felk.cvut.cz/courses/PGR/doxygen.html`

[28] van Heesch, D.: *Doxygen: Generate documentation from source code [online]*. 2015, [cit. 2015-04-06]. Dostupné z: `http://www.doxygen.nl/`

[29] Olšák, P.: *Lineární algebra [online]*. Praha, druhé vydání, 126 s., definice 13.23. Dostupné z: `http://math.feld.cvut.cz/olsak/ftp/olsak/linal/linal2.pdf`

[30] Google Inc.: *Google mapy [online]*. 2015, [cit. 2015-03-23]. Dostupné z: `https://maps.google.com`

[31] Sedláček, L.: *SAGE2 Contactless control [online]*. 2015, [cit. 2015-04-16]. Dostupné z: `https://bitbucket.org/sedlalu2/sage2-contactless-control`

Příloha A

Seznam použitých zkratek

API Application Programming Interface

EVL Electronic Visualization Laboratory

FPS Frames Per Second

GUI Graphical User Interface

HTML HyperText Markup Language

ID Identification

IP Internet Protocol

iOS Mobilní software společnosti Apple

SAGE|SAGE1 Scalable Adaptive Graphics Environment

SAGE2 Scalable Amplified Group Environment

SAGElab Síťová a multimediální laboratoř ČVUT

SDK Software Development Kit

TCP Transmission Control Protocol

URL Uniform Resource Locator

USB Universal Serial Bus

Wi-Fi Wireless LAN

Příloha **B**

Obsah přiloženého CD

- `readme.txt` — stručný popis obsahu CD
- `samples` — videoukázky z použití aplikace
  - `author.mp4` — ovládání autorem aplikace
  - `user_Mosna_David.mp4` — ovládání testovaným uživatelem
- `src`
  - `prototype` — prototyp aplikace se zdrojovými kódy
    - `bin` — adresář pro umístění vygenerované binární podoby
    - `config` — umístění konfiguračních souborů
    - `doc` — adresář pro vygenerování dokumentace
      - `images` — obrázky použité v dokumentaci
    - `lib` — knihovny použité v implementaci
      - `easywsclient`
      - `NiTE-Linux-x64-2.2`
      - `OpenNI`
    - `src` — zdrojové kódy prototypu
  - `thesis` — zdrojová forma práce ve formátu LaTeX
    - `images` — obrázky použité v literární rešerši
- `text` — text práce
  - `BP_Sedláček_Lukáš_2015.pdf` — text práce ve formátu PDF