VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

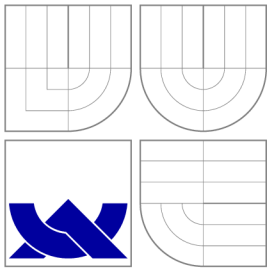
KRESLENÍ VE 3D POMOCÍ LEAP MOTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

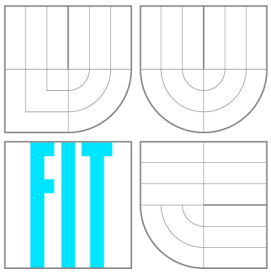
AUTOR PRÁCE
AUTHOR

JAKUB SEMERÁK

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KRESLENÍ VE 3D POMOCÍ LEAP MOTION

3D DRAWING USING LEAP MOTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAKUB SEMERÁK
AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL NAJMAN
SUPERVISOR

BRNO 2014

Abstrakt

Cílem této práce bylo vytvoření jednoduché aplikace pro 3D kreslení pomocí senzoru Leap Motion. Na této aplikaci byla testována efektivita interakce s rozhraním gesty oproti interakci kombinované s klávesou. Testováním bylo zjištěno, že ovládání kreslení gestem není tak efektivní jako klávesou. Nicméně po delší interakci uživatelů s aplikací se projevilo znatelné zrychlení ve prospěch aktivace gestem.

Abstract

The goal of this thesis was to create a simple 3D drawing application that uses Leap Motion controller. This application was designed to test gesture-based interaction versus keyboard controls. It was found that gesture-based interaction is not as effective as interaction combined with keyboard. However, when users interact with the application for a prolonged period of time, the noticeable acceleration of gesture-based interaction could be observed.

Klíčová slova

rozpoznávání gest, Leap Motion, počítačové vidění, 3D kreslení, virtuální rozhraní

Keywords

gesture recognition, Leap Motion, computer vision, 3D drawing, virtual interface

Citace

Jakub Semerák: Kreslení ve 3D pomocí Leap Motion, bakalářská práce, Brno, FIT VUT v Brně, 2014

Kreslení ve 3D pomocí Leap Motion

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Najmana.

.....
Jakub Semerák
20. května 2014

Poděkování

Tímto bych rád poděkoval panu Ing. Najmanovi za odborné vedení práce a za jeho věcné rady a připomínky.

Obsah

Kapitola 1

Úvod

Leap Motion je poměrně nové zařízení umožňující sledování lidského těla. Soustředí se především na rozpoznávání gest, sledování rukou a prstů člověka. Díky tomuto senzoru si můžeme prakticky vyzkoušet alternativní způsob ovládání například počítače pouhým pohybem rukou prostorem. Samotný senzor funguje na principu počítačového vidění, ze kterého se snaží rozpoznat pozici uživatelových rukou v prostoru na základě zpracování obrazu. Jedná se vlastně o jeden z prvních běžně dostupných senzorů se zaměřením čistě na detekci rukou pomocí počítačového vidění.

Tato práce probírá základní problematiku rozpoznávání gest pomocí zpracování obrazu a problémy s touto metodou úzce související. Poté vzájemně srovnává dostupné senzory a jejich detekční schopnosti, kde hlavním kritériem je přesnost senzorů.

Hlavním cílem práce je vytvoření jednoduché demonstrační aplikace, umožňující kreslení ve 3D prostoru. Po teoretickém úvodu se nejprve práce zaměřuje na návrh, kde je navrženo rozhraní aplikace a jejích funkcí. Po implementaci aplikace je dalším krokem ověření spolehlivosti ovládání čistě pomocí rukou a gest otestováním na skupině uživatelů. Efektivita tohoto způsobu ovládání je srovnávána s alternativním ovládáním částečně využívajícím klávesnici. Závěr práce se následně věnuje vyhodnocení dotazníku a provedeného testování.

Kapitola 2

Rozpoznávání gest

Jelikož je tato práce zaměřena na sledování rukou uživatele, je pro nás zásadní jeho interakce s prostředím aplikace jednoduchými gesty. A právě tomu se tato kapitola věnuje. V první podkapitole 2.1 je rozebrán význam samotných gest a metody jejich rozpoznávání. Následně je nutné zpracovat vstupní data ze senzoru, který aktivně sleduje ruce uživatele a odesílá informace o pohybu ruky a jejích částí prostorem v čase. K tomuto je využit senzor *LEAP Motion*[5], který byl vybrán pro realizaci této práce a srovnáván s obdobnými senzory běžně dostupnými senzory v podkapitole 2.3.

2.1 Gesta a jejich detekce

Jak uvádí Matthew Turk[13], koncept gesta je poměrně volně definován a velmi záleží na kontextu jeho použití. Jedním z hlavních problémů je odlišení gest od ostatních interakčních pohybů. Obecně není žádný standardní způsob, jak rozpoznávat gesta, nicméně většina systémů na rozpoznávání gest funguje na podobném principu.

Základním typem jsou *statická gesta*, kdy je gesto reprezentováno momentální zaujatou pózou ruky. Mezi tyto gesta patří například „palec nahoru“, „rozevřená ruka“ apod. Za druhý typ gest jsou považována *dynamická gesta*, kdy gesto definuje pohyb objektu prostorem. Například McNeill[7] člení dynamická gesta do tří skupin. První z nich je stav před tahem (*pre-stroke*), následuje tah (*stroke*) a nakonec stav po tahu (*post-stroke*). Některé z gest se mohou skládat ze statických i dynamických částí, kdy póza např. ruky je jedna z fází gesta. Zvažovat lze i závislost na gestech přímo předcházejících či následujících před aktuálním gestem.

Hummells a Stappers[2] nastiňují následující čtyři základní rysy gest, které jsou důležité pro jejich rozpoznání.

- *Prostorová informace* (angl. Spatial) - místo, kde se gesto vyskytlo a k jakému místu se vztahuje.
- *Informace o cestě* (angl. Pathic) - trasa, kterou gesto zabírá.
- *Znaková informace* (angl. Symbolic) - tvarová informace o cestě.
- *Ráz gesta* (angl. Affective) - rychlost, dynamičnost gesta.

Pro rozpoznání těchto rysů gesta je nutné především zpracovávat informace o pozici, rychlosti pohybu a zaujatém postavení (úhlů jednotlivých částí těla vůči sobě). Tyto informace se pro výzkumné účely získávají pomocí datových rukavic, datových obleků apod.

Pro běžné použití jsou určeny senzory založené na počítačovém vidění, jak je rozebíráno v kapitole 2.3. Tyto varianty se odlišují zejména přesností, prodlevou, komfortem pro uživatele a v neposlední řadě cenou, která bývá hlavně u datových obleků poměrně vysoká. U statických gest se jejich rozpoznání provádí pomocí poměrně přímočarých metod (například porovnávání pomocí šablon, neuronové sítě). Rozdělení rozpoznávání gest do fází vybízí k využití konečných automatů, kdy při rozpoznání předem stanoveného postavení ruky začne například trasování pohybu ruky.

2.2 Sledování lidského těla

V praxi se využívají dva základní způsoby sledování lidského těla pro rozpoznání gest. Ty srovnává Matthew Turk ve svojí práci[13], ve které mimo jiné popisuje, čeho by se měli vývojáři držet při návrhu rozhraní využívající gesta. První z těchto způsobů jsou metody založené na sledování pohybu ruky pomocí senzorů připevněných na těle (například datové rukavice – angl. instrumented gloves). Jejich velkou nevýhodou je právě nutnost uživatele oblékat si na své tělo oblek obsahující sledovací zařízení. Dále bývá u těchto zařízení složitá kalibrace a je nutné poměrně velké investice do kvalitních senzorů, aby z nich získaná data byla na potřebné úrovni. Naopak jejich výhodou jsou přesné informace o poloze ruky, úhly mezi částmi ruky, aktuální rotace a sklon zápěstí atd.

Tato práce se zabývá právě druhým způsobem, který je založen na rozpoznání lidského pohybu z obrazu v reálném čase. I u tohoto způsobu se v některých případech používají například značky na sledovaném těle nebo speciální obleky s označením důležitých částí těla, aby došlo k usnadnění a upřesnění rozpoznávaných částí lidského těla v obraze. Právě tato metoda projevuje jistý potenciál budoucího širšího využití při interakci s virtuálním prostředím. Jejím cílem je, aby sledování lidského pohybu bylo v budoucnu co nejpřesnější a nebylo tedy zapotřebí používat nepohodlné obleky či značky na lidském těle.

Zařízení, která se zaměřují na poslední zmíněný způsob používají jednu nebo více kamer, kterými sbírají snímky lidského těla při frekvenci alespoň 30 Hz či více. Po zpracování těchto snímků interpretují získaná data jako informace o pozici prstů ruky, směru dlaně apod. Většinou jsou kamery umístěny ve scéně staticky, v některých případech na pohyblivém podstavci nebo na jiné osobě.

Na rozdíl od senzorů umístěných na těle se metoda založená na počítačovém vidění musí potýkat s poměrně významným problémem, a to *překrýváním*. Z pohledu kamery se vždy najde část uživatelova těla, která je zastíněna a tudíž není pro kameru viditelná (angl. *self-occlusions*). To ztěžuje například rozpoznání umístění všech prstů nebo přesné postavení ruky z jednoho pohledu kamery. Řešením tohoto problému by mohlo být rozmístění více kamer z dalších úhlu, čímž ale vyvstanou další problémy k řešení, jako například synchronizace kamer, finanční náročnost a vyšší nároky na přesné vzájemné rozmístění senzorů. Problém překrývání velmi ztěžuje až znemožňuje přesné sledování všech částí těla zároveň. Tento problém však nemusí mít přímo vliv na rozpoznávání gest a také se nám nabízí řešit tento problém některým z následujících způsobů. Parametry ruky, které nejsou aktuálně k dispozici:

- mohou být dopočítány,
- nejsou nezbytně potřebné k dokončení úlohy,
- nebo jsou některé dopočítány a ty nepotřebné zanedbány.

Dalším problémem u této metody může být na rozdíl od senzorů na těle nedostatečná rychlost algoritmu pro rozpoznání částí těla z obrazu při rychlém pohybu a tudíž nízká přesnost zařízení. To je například věc, kterou u běžně dostupných komerčních senzorů nelze ovlivnit, jelikož si většinou výrobce technologické detaily a algoritmy chrání. Hlavními kritérii při volbě senzoru pro tuto práci byly dostatečná rychlost zpracování pohybu rukou bez zbytečné prodlevy a přesnost při zpracování polohy ruky pro pohodlnou manipulaci s prostředím.

2.3 Srovnání senzorů

Dnes se na trhu vyskytuje poměrně velké množství senzorů, umožňující sledování lidské postavy v reálném čase. Některé z nich jsou však určeny spíše pro sledování celé lidské postavy a nejsou přímo vhodné pro přesné a rychlé sledování rukou v reálném čase. To je blíže popsáno v následujících podkapitolách, ve kterých jsou srovnávány senzory dnes běžně dostupné pro širší veřejnost. Nejprve jsou zmíněny některé z prvních typů senzorů pro ovládání zařízení pohybem. Následují pro tuto práci důležitější a novější zařízení, využívající výhradně počítačové vidění.

2.3.1 Nintendo Wii

Tento systém od firmy Nintendo[9], jehož prodej byl zahájen ke konci roku 2006, se dá považovat za jistého průkopníka v technologiích určených pro interakci s rozhraním pomocí pohybu uživatele. Celá sada ideálně využívá pro detekci pohybu dva interaktivní ovladače (Wii Remote a Nunchuk zobrazené na obrázku 2.1), které jsou vzájemně spojeny kabelem. Tyto ovladače jsou vybaveny akcelerometrem, který je schopen detekovat zrychlení a směr pohybu ve všech třech osách. Dále je doplněna o infračervenou kameru, která určuje pozici ovladačů pomocí sledování LED diod zabudovaných v ovladačích. V poslední řadě je možné do celého setu zakomponovat tlakovou podložku Wii Balance Board, která monitoruje postoj člověka a jeho případný přesun váhy. Veškeré zachycené informace jsou bezdrátově pomocí rozšířené Bluetooth[1] technologie přenášeny k cílovému zařízení a dokonce je tedy možné přímo připojit Nintendo Wii k různým druhům zařízení včetně osobních počítačů.

Obrázek 2.1: Sada ovladačů – vlevo Wii Nunchuk a vpravo Wii Remote[9]

Kombinací výše zmíněných prvků je možné dosáhnout relativně přesných informací o pohybu člověka a umožňuje tak celkem spolehlivé rozpoznávání prostorových gest, což

[1] www.bluetooth.org

také ulehčují zabudované akcelerometry v ovladačích. Konkrétně lze sledovat postoj a přesun váhy postavy (pomocí tlakové podložky), dynamické pohyby rukou (pomocí dvojice ovladačů) a jejich polohu (pomocí kamery). Nejsou tedy dostupné informace o přesném postavení těla uživatele a při použití ovladačů držených v rukou samozřejmě nelze využít ani prsty rukou. Právě proto není pro naše účely není příliš vhodný, ale i tak lze Nintendo Wii poměrně efektivně využít pro ovládání rozhraní gesty, kde bližší přehled o prstech není zapotřebí. Jako celek je toto zařízení poměrně jednoduché a díky tomu je i relativně levné (zaváděcí cena byla v USA necelých 250 USD[2]) a nejspíše i přimělo ostatní hlavní konkurenty (Microsoft, PlayStation) jednat a vyvinout vlastní zařízení.

2.3.2 PlayStation Move

Bezdrátový ovladač PlayStation Move byl na trh uveden v září 2010 firmou Sony[11]. Opět se jedná o bezdrátový ovladač (využívající technologii Bluetooth) v kombinaci s vysokorychlostní kamerou, která dosahuje rychlosti od 60 do 120 Hz na základě použitého rozlišení. Ovladač je zakončen kulatou částí, která vyzařuje světlo v celém pásmu RGB. Barva této části se mění dynamicky v závislosti na prostředí tak, aby byla co nejvíce kontrastní s prostředím a ulehčilo se tím následné zpracování. Případně mohou být využity zároveň až 4 ovladače (každý s jinou vyzařovanou barvou). PlayStation Move poté dokáže prostřednictvím kamery s pevným ohniskem, předem známé velikosti objektu a aktuální vyzařované barvě určit poměrně spolehlivě pozici ovladače v prostoru i za zhoršených světelných podmínek. Celá sada je zobrazena na obrázku 2.2.

Ovladač je také doplněn o zabudované senzory v ovladači, které napomáhají vyhodnocení pohybu ovladače. Konkrétně je v ovladači umístěn tříosý gyroskop pro určení náklonu, tříosý akcelerometr pro určení směru pohybu a magnetometr pro přesnější určení polohy podle magnetického pole Země. Tyto senzory usnadňují určení pozice ovladače a umožňují alespoň přibližné určení pozice, pokud senzor není zrovna pro kameru viditelný (například je zrovna překryt jiným objektem). Opět je zde uživatel omezen nutností držet v rukou ovladače a faktem, že má PS Move přehled pouze u pozici ovladačů a ne o přesném postavení těla uživatele. Tento fakt však neomezuje celkem spolehlivé rozpoznání pohybu rukou uživatele, ale bez sledování jeho prstů. Informace o ovladači byly čerpány z online materiálů firmy Sony[11, 12].

Obrázek 2.2: Sada Playstation Move – kamera, ovladač a samotné zařízení[11]

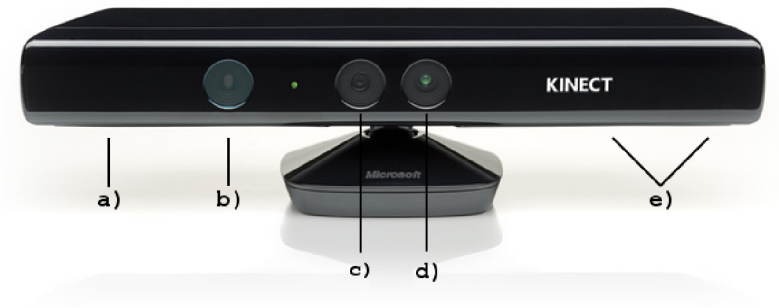
[2]Zdroj: http://www.ign.com/articles/2006/09/14/us-wii-price-launch-date-revealed

2.3.3 Microsoft Kinect

Microsoft Kinect je pohybový senzor, který byl z počátku dodáván jako součást balení herní konzole Xbox 360. V červnu 2011 však bylo vydáno pro nekomerční využití Kinect SDK for Windows[8] určené pro počítače s Windows 7 (v únoru 2012 Microsoft vydal toto SDK i pro komerční užití). Tímto se Kinect oficiálně dostal od konzolí do odvětí osobních počítačů.

Kinect je osazen RGB kamerou, která je schopná zaznamenávat obraz v rozlišení 640x480 pixelů při frekvenci 30 Hz (při frekvenci 12 Hz uchovává snímky o rozlišení 1280x960 pixelů). Dále obsahuje infračervený projektor, jež směrově vysílá do scény infračervené paprsky, jejichž odrazy následně zachycuje infračervený hloubkový senzor (infračervená kamera s rozlišením 640x480 pixelů). Z informací o zachycených paprscích si Kinect dokáže dopočítat vzdálenost každého bodu ve scéně, kde dopadl paprsek. Dále obsahuje čtyři vzájemně propojené mikrofony, pomocí kterých dokáže určit, z jakého směru dorazil zvuk k zařízení. Výše zmíněné komponenty senzoru jsou znázorněny na obrázku 2.3.

Tento senzor je primárně určen pro snímání celé postavy člověka ve větších vzdálenostech od senzoru a není tedy příliš vhodný v tomto případě pro interakci s 3D prostorem. V oficiálním SDK není v základu podporováno sledování prstů ruky, i když jsou volně ke stažení dostupné neoficiální knihovny, které tuto funkčnost částečně podporují[3].



Obrázek 2.3: Microsoft Kinect – a) Mikrofon; b) Infračervený projektor; c) RGB kamera; d) Infračervený hloubkový senzor; e) Další tři mikrofony[8]

Na trhu se však již objevila nová verze Microsoft Kinect, která vyšla společně s konzolí Xbox One 22. listopadu 2013. Bohužel je oficiálně kompatibilní pouze s konzolí Xbox One. Vydání verze pro PC je očekáváno během roku 2014.

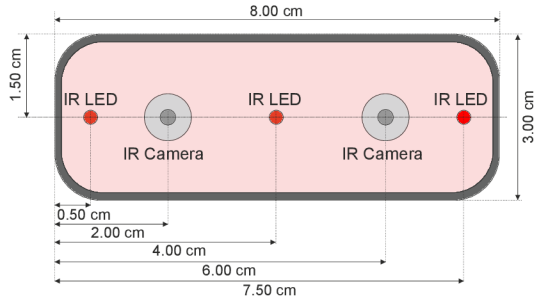
2.3.4 LEAP Motion Controller

Jedná se o nedávno vydané USB zařízení od firmy Leap Motion, Inc[5] primárně určené pro rozpoznávání gest rukou a pozice prstů v prostoru v interaktivních aplikacích. První verze zařízení se k uživatelům dostala v červenci 2013 a od této doby jsou zařízení rozesílána do celého světa. Tento senzor je pro nás v této práci nejpodstatnější, proto mu je věnována větší pozornost. Jelikož je však toto zařízení proprietární, neexistuje k němu žádná oficiální podrobná dokumentace. Z toho důvodu je nutné vycházet z obecnějších popisů zařízení a výrobcem nepotvrzených informací.

V následujících odstavcích je probírána nejprve hardwarová stránka senzoru, kterou následují způsoby zpracování informací ze senzoru.

[3]Například http://www.kinecthacks.com/kinect-hand-tracking-gesture-experiment.

Hardwarová část Na rozdíl od zařízení Microsoft Kinect nevyužívá při detekci hloubkový senzor, ale skládá se ze dvou infračervených kamer a tří infračervených vysílačů (což jsou vlastně infračervené LED diody) jak naznačuje schéma na obrázku 2.4. Na základě právě těchto dvou kamer je možné Leap Motion zařadit do kategorie optických sledovacích systémů založených na tzv. *stereo vision*, volně přeloženo jako stereo vidění. Princip této metody spočívá v nalezení společných bodů ve 2D obrázcích ze dvou kamer, u kterých jsou známé jejich vzájemné parametry (například vzájemná pozice kamer).



Obrázek 2.4: Schéma senzoru Leap Motion Controller[14]

V této studii[14] byla sérií měření provedena analýza přesnosti a robustnosti tohoto senzoru. Na základě faktu (uvedeného v této studii), že lidská ruka je v průměru schopna dosáhnout přesnosti 0,4 mm, byl v této studii použit průmyslový robot, který je schopen pohybovat svým ramenem prostorem s přesností 0,2 mm. V sérii statických testů, kdy byl sledovaný objekt vůči senzoru nehybný, byla odchylka mezi reálnou a změřenou pozicí objektu pro všechny osy méně než 0,2 mm (výsledky v závislosti na průměru sledovaného objektu se téměř nelišily). V sérii dynamických testů, kdy pohyboval robot tužkou vůči senzoru prostorem po stejné dráze, byla směrodatná odchylka pro všechny osy menší než 0,7 mm.

Přestože se nepodařilo dosáhnout teoretické přesnosti uváděné výrobcem (0,01 mm[4]), má Leap Motion Controller s celkovou průměrnou přesností do 0,7 mm celkem slibné vyhlídky pro přesné sledování pohybu rukou uživatele. Například v cenově podobné kategorii senzor Microsoft Kinect dosahuje směrodatné odchylky v přesnosti přibližně 1,5 cm, jak je uváděno v této práci[3].

V současné době by se tedy mělo jednat o jedno z nejpřesnějších cenově přijatelných řešení pro sledování pohybu rukou.

Softwarová část Ze softwarové části je především zajímavé výrobcem poskytované rozhraní umožňující programování aplikací s využitím sbírky funkcí či procedur konkrétních knihoven (dále v textu jako API – Application Programmer Interface). Rozhraní Leap API je dostupné ve dvou formách, kdy první a zároveň pro tuto práci nejdůležitější z nich je dynamická knihovna, která umožňuje práci s funkcemi senzoru. Druhou variantou je *WebSocket* rozhraní, které slouží pro vytváření webových aplikací. API je dostupné pro většinu dnes běžných platforem – Windows, OSX a Linux. Senzor ve spojení s aktuálním API poskytuje vývojářům souřadnice předdefinovaných objektů nad senzorem v pravorukém kartézském souřadnicovém systému, jak je znázorněno na obrázku 2.5. Hodnoty souřadnic jsou uvedeny v reálných milimetrech relativně vzhledem ke středu senzoru. Během sledování objektů senzor poskytuje zachycené informace po datových snímcích (poskytováno

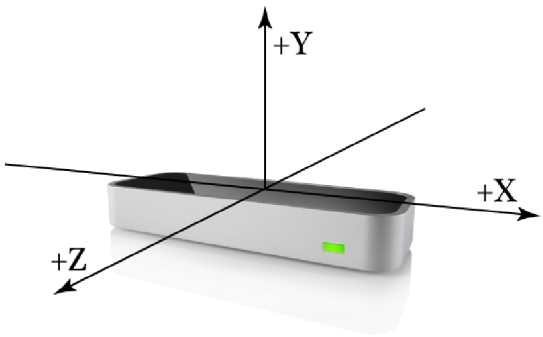
[4]Z oficiální stránky produktu: https://www.leapmotion.com/product

API jako objekt `Frame`). Každý snímek (*frame*) obsahuje seznamy s rozpoznanými objekty, gesty a dalšími faktory. Následuje výčet objektů, které obsahují seznamy v objektu snímku.

- `Hand` – Veškeré ruce rozpoznané senzorem (obsahuje informace například o pozici ruky, její aktuální rychlost, směrový vektor z dlaně).
- `Finger` – Všechny prsty náležící k nějaké ruce (`Hand`).
- `Tool` – Veškeré rozpoznané nástroje, co nejsou prsty.
- `Pointable` – Ukazovací objekt je abstrakcí pro prsty a nástroje (objekty `Finger` a `Tool`). Obsahuje informace o délce a výšce objektu, směrový vektor a souřadnice a rychlost špičky objektu.
- `Gesture` – Veškerá gesta, která začala, skončila nebo se aktualizovala – například gesto kruh (*circle*), posun (*swipe*) či stisknutí klávesy (*key tap*) nebo stisknutí na obrazovce (*screen tap*).

Každému rozpoznanému objektu (dokonce i gestu) API přiřadí unikátní identifikátor, který zůstává nezměněný po celou dobu pohybu ve scéně. Pokud objekt opustí scénu, není zaručeno, že mu bude po jeho opětovném objevení ve scéně přiřazen stejný identifikátor (senzor nemusí rozpoznat, že se jedná o tentýž objekt).

Leap API v základu poskytuje pět operačních módů, které přímo ovlivňují způsob analyzování dat ze senzoru. První tři módy patří do *primární* skupiny, počínaje *precizním módem*, který upřednostňuje přesnost před rychlostí (cca 80 snímků za sekundu). Druhý *vysokorychlostní mód* poskytuje cca 295 snímků za sekundu na úkor přesnosti. *Balancovaný mód* je potom kompromisem mezi předchozíma dvěma módy (cca 150 snímků za sekundu) a i ve výchozím nastavení používaným módem. Uvedené rychlosti platí pro využití USB 3.0. Při použití USB 2.0 je počet snímků omezen nižší rychlostí rozhraní. Zbývající módy jsou *sekundární* a patří mezi ně *robustní mód*, jehož aktivaci nemůže uživatel přímo ovlivnit, protože dochází k jeho aktivaci samovolně v závislosti na světelných podmínkách a *úsporný mód*, který snižuje energické nároky, vytížení CPU a datový tok.



Obrázek 2.5: Pravotočivá soustava kartézských souřadnic vzhledem k senzoru[5]

2.3.5 Srovnání

Cílem této kapitoly bylo nastínění základních rysů běžně používaných senzorů a srovnání jejich vlastností. PlayStation Move a Nintendo Wii jsou jedny z prvních senzorů, které

nastolily směr pro vývoj jejich následníků a vzhledem k jejich omezenějším možnostem v přesném sledování rukou nejsou pro naši práci úplně vhodné. Daleko lepší je pro naši práci senzor Leap Motion, který se přímo soustředí na sledování pozice rukou a prstů v prostoru. Také se jedná o relativně nové zařízení, jehož API je neustále ve vývoji, vývojáři ho nadále zdokonalují a pravděpodobně se v něm skrývá velký potenciál. Poslední z porovnávaných zařízení Microsoft Kinect umožňuje sice univerzálnější použití, ale nedisponuje přesností Leap Motion senzoru. Nezbývá než vyčkat, co přinese jeho vylepšená verze, která bohužel nebyla v době psaní práce dostupná pro odvětví osobních počítačů.

Kapitola 3

Návrh rozhraní a funkcí aplikace pro kreslení

V této kapitole jsou specifikovány hlavní rysy rozhraní a funkcí aplikace, pomocí které je zkoumána spolehlivost manipulace uživatelů s virtuálním prostředím. Aplikace by měla poskytnout praktický příklad interakce s rozhraním čistě za pomoci gest. Počítáno je také s úpravami rozhraní aplikace na základě výsledků testování, a to i za cenu využití klávesnice jako dalšího vstupního zařízení.

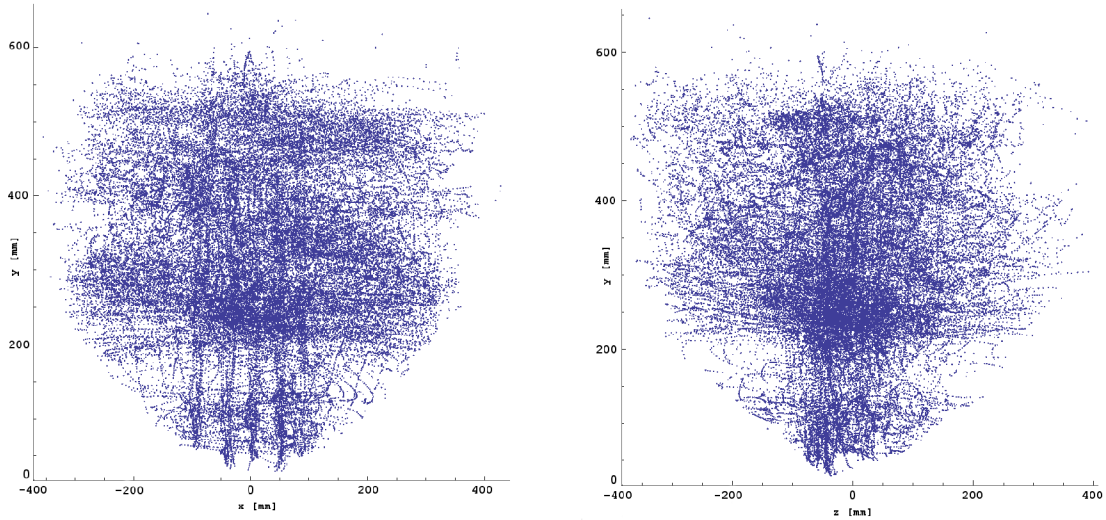
Pomocí aplikace se budou vytvářet jednoduché různobarevné trasy ve 3D prostoru, přičemž bude možné ovlivnit způsob vykreslení trasy i její průměr.

Nejprve je v podkapitole 3.1 proveden prvotní návrh prostředí aplikace, od kterého se odvíjí další části návrhu. V následující podkapitole je rozebrán návrh struktury a principu výběru položek z menu. Další podkapitola 3.3 popisuje návrh interakce pomocí gest a na závěr podkapitola 3.4 nastiňuje potenciální problémy, které se mohou při implementaci objevit.

3.1 Návrh prostředí

Rozložení pracovního prostoru je jedna z prvních věcí, na kterou se zaměříme. V potaz je nutno brát hlavně rozsah Leap Motion senzoru, ze kterého se budou získávat data o pozici rukou uživatele. Jeho detekční schopnosti se liší v závislosti na vzdálenosti sledovaného objektu od senzoru, a proto je důležité zvolit z tohoto rozsahu správnou oblast, kde je prostor pro manipulaci co největší a především v přijatelné vzdálenosti od senzoru. Z tohoto důvodu jsem provedl sběr souřadnic ze senzoru, jehož výstup je zobrazen v grafech na obrázku 3.1. Na obou grafech vidíme ortogonální průmět bodů na příslušnou rovinu. První graf 3.1a reprezentuje pohled na nasbírané souřadnice zepředu promítnutý na rovinu xy. Druhý graf 3.1b je pohled zleva promítnutý na rovinu yz. Jako vstupní data bylo nasbíráno přibližně **75 tisíc** souřadnic, aby byl zmapován co možná největší reakční prostor senzoru.

Na základě výstupu předchozího testování jsem se rozhodl umístit potenciální kvádr, jež reprezentuje reakční *pracovní prostor* senzoru, do intervalu $< -200, 200 >$ pro horizontální osy x a y a do intervalu $< 150, 550 >$ pro vertikální osu z. Aplikace však bude mít přehled o prstech i mimo tento prostor, kreslení mimo něj však nebude možné. Tento *pracovní prostor* je znázorněn na obrázku 3.2. Jeho velikost se bude přizpůsobovat v závislosti na použitém rozlišení při spuštění aplikace a poměru stran vždy tak, že nejširší strana rozlišení bude roztažena do úplného rozsahu senzoru a zbylý rozměr bude zabírat



(a) (b)

Obrázek 3.1: a) Pohled zepředu vůči senzoru; b) Pohled zleva vůči senzoru.

rozsah senzoru odpovídající poměru mezi stranami rozlišení. Výsledný kvádr tedy bude pravidelným čtyřbokým hranolem, jehož základna (tj. šířka a hloubka) bude mít rozměry odpovídající horizontálnímu rozlišení. Zbývající rozměr hranolu – jeho výška – je tedy dána vertikálním rozlišením okna aplikace.

vertikální rozlišení

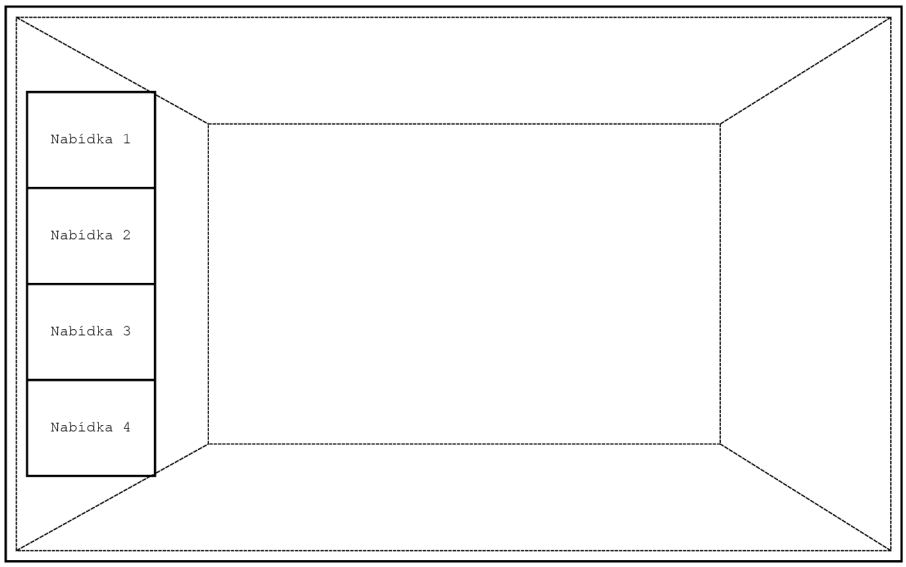
horizontální rozlišení

Obrázek 3.2: Pravidelný čtyřboký hranol reprezentující pracovní plochu

Ve výchozím stavu po spuštění aplikace se uživatel ocitne přímo v 3D pracovním prostoru, kde případně bude moci aktivovat menu a zvolit některou z dostupných akcí nebo rovnou zahájit proces kreslení. Pokud uživatel umístí ruku nad senzor, dojde k vybrání jednoho z prstů ruky a zobrazení jeho aktuální polohy umístěním objektu ve tvaru koule do 3D prostoru – dále v textu pod pojmem *ukazatel*. Způsob výběru akce z menu bude z počátku čistě pomocí gest a pohybu ruky, kdy až následným testováním případně aplikujeme potřebné změny. Návrh menu je blíže popsán v následující kapitole.

3.2 Návrh struktury a umístění menu

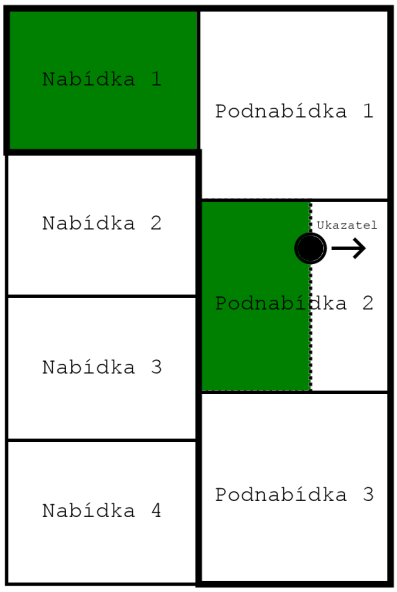
Celá aplikace bude ovládána především skrze menu, které bude umístěno v popředí v levé části obrazovky. Umístění menu je zobrazeno na obrázku 3.3. Během kreslení bude skryto a k jeho zobrazení dojde, jakmile uživatel provede gesto pro zobrazení menu. Tento způsob manipulace s rozhraním aplikace byl zvolen, protože se jedná o dnes běžnou metodu použitou snad v každém klasickém 2D prostředí, a tudíž většina uživatelů bezpečně rozpozná tento prvek jako menu. Nevýhodou by mohlo být zakrytí určité části zorného pole tímto menu, čemuž bude zabráněno jeho skrytím po výběru položky.



Obrázek 3.3: Návrh umístění menu na obrazovce

V tomto případě bude menu dvouúrovňové, kdy položky budou rozděleny do skupin, ze kterých bude uživatel vybírat. Po zobrazení menu se na obrazovce zobrazí *ukazatel*, zobrazující místo, na které ukazuje uživatel vztyčeným prstem (případně se vybere nejdelší z detekovaných prstů). Bude se tedy provádět projekce směrového vektoru na rovinu obrazovky. Umístěním kurzoru nad položku první úrovně se zobrazí druhá úroveň. Prvotní návrh počítal s rozbalením položek v horizontálním směru místo dále popsaného vertikálního rozbalení. U něj by však bylo složitější znázornění větvení dalších úrovní a implementace a pohyb v menu by nabral na složitosti.

Výsledný návrh a proces výběru z menu je zobrazen na obrázku 3.4, kdy byl nejprve umístěn kurzor nad položku *Nabídka 1*, čímž došlo k zobrazení druhé úrovně. K vybrání položky *Podnabídka 2* z druhé úrovně poté dojde jejím přetažením pomocí kurzoru směrem doprava až za pravou hranu menu. Míra přetažení a tedy i zbývající dráha k výběru položky je naznačena zeleným podbarvením části položky menu. Kurzor je znázorněn černou tečkou s šipkou značící směr pohybu pro úspěšné vybrání položky z menu. V každé úrovni se položky roztáhnou na plnou výšku menu, aby bylo dosaženo co možná největší plochy položek a nedošlo tedy omylem k vybrání jiné položky. Návrh menu se snaží držet doporučení pro vývojáře uvedených v článku[6] o osvědčených druzích menu vhodných pro senzor Leap Motion.



Obrázek 3.4: Způsob výběru položky z menu

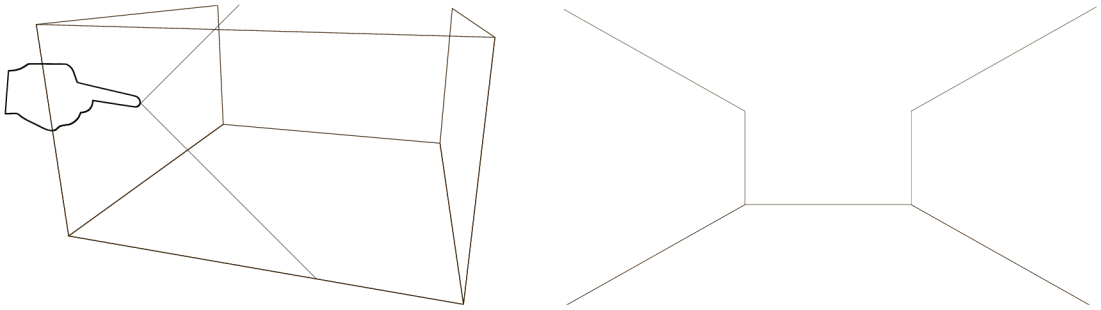
Dostupné položky menu Zde je popsán obsah postranního 2D menu, což je zároveň i seznam nastavitelných parametrů aplikace. Menu bude sloužit zejména k nastavení parametrů kreslení či případně ke změně pohledu apod. Samotná aktivace kreslení nebo manipulace se scénou bude ovládána pomocí gest, tomu se ale věnuje až další podkapitola 3.3.

Výběr velikosti nástroje V postranním 2D menu bude možné změnit velikost nástroje, se kterým se bude kreslit. To ve výsledku ovlivní šířku trasy, jež se během kreslení vytvoří. Bude možné vybírat ze tří předdefinovaných velikostí.

Výběr barvy nástroje Obdobě jako velikost nástroje bude možné ovlivnit barvu vytvořené trasy, kdy bude uživatel vybírat z předdefinovaných barev. K dispozici budou čtyři předdefinované barvy, a to žlutá, zelená, modrá a červená.

Rozhlížení Mód rozhlížení slouží v první řadě k prohlédnutí již hotové scény, nebo případně k náhledu pod konkrétním úhlem. K tomu bude použita technika „*kamera v prstu*“ (viz *Camera-in-Hand Technique*[1]), kdy je virtuální kamera umístěna v prstu uživatele jako na obrázku 3.5a. Uživatel může volně pohybovat kamerou 3D prostorem a jako výstup mu bude zobrazena scéna z pohledu prstu směřujícího do *pracovního prostoru*, což je vidět na obrázku 3.5b. Tento způsob však může být pro uživatele matoucí, jelikož je scéna vykreslována z pohledu první osoby, přičemž on ji vidí z pohledu třetí osoby. Jak se tento způsob osvědčí v praxi bude ověřeno pozdějším testováním.

Typ kreslení Aplikace bude umožňovat dva odlišné typy vykreslování trajektorie pohybu *ukazatele*. První z nich při každé aktualizaci pozice *ukazatele* vykreslí přesně na jeho místě kouli podle přednastavených parametrů tj. (barva, velikost). Druhým z nich je, že se zaznamenané polohy budou mezi sebou spojovat vykreslením válce a díky tomu by měla být trajektorie mnohem plynulejší.



(a) (b)

Obrázek 3.5: a) Technika „*kamera v prstu*“; b) Pohled z virtuální kamery do scény.

Vymazání scény Tato operace jednoduše vymaže veškeré nakreslené útvary z *pracovního prostoru*.

Skrytí menu V poslední řadě bude v 2D menu prvek pro opuštění menu, po jehož vybrání dojde pouze ke skrytí postranního menu.

3.3 Interakce gesty

V této podkapitole jsou popsány způsoby, kterými uživatel bude gesty předávat příkazy přímo aplikaci bez nutnosti aktivace hlavního menu.

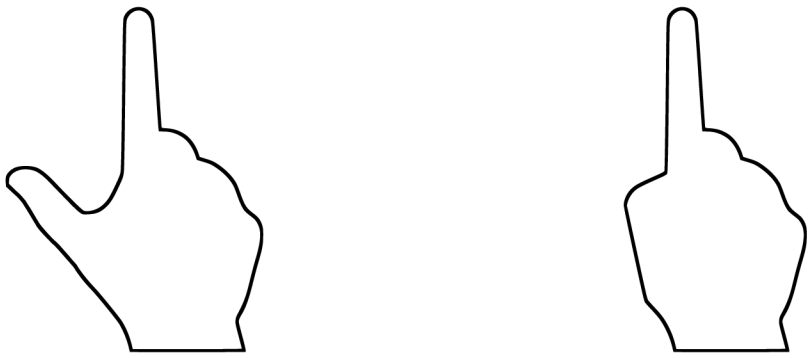
Manipulace s pracovním prostorem Manipulace s 3D scénou při skrytém menu a neaktivním kreslení bude možná pomocí jednoduchého *swipe* gesta, které otočí scénu horizontálně o 90 stupňů doleva nebo doprava. Směr otočení vyplývá ze směru provedeného gesta a pro představu je samotné gesto znázorněno na obrázku 3.6, který je převzatý z oficiální dokumentace.

Obrázek 3.6: Gesto *swipe* podporované Leap Motion API[5]

Kreslení a ukazování Do módu kreslení se uživatel dostane z režimu *ukazování*, jež bude aktivní ihned po spuštění aplikace. V režimu *ukazování* může uživatel pohybovat

ukazatelem po scéně, aniž by se za nástrojem vykreslovala jeho trajektorie. *Ukazatelem* rozumíme prst, který byl prohlášen za nejdelší a určuje pozici grafického prvku znázorňující *ukazatel* ve scéně. Ideální postavení prstů ruky pro bezpečné rozpoznání postavení ruky jako *ukazatele* je patrné z obrázku 3.7b. K zahájení vykreslování trajektorie *ukazatele* dojde po jeho aktivaci, a to konkrétně vztyčením prstu (ideálně palce) tak, aby prst, který je rozpoznán jako *ukazatel*, svíral s nově vztyčeným prstem úhel větší nebo rovno 30 stupňů a zároveň byly na ruce rozpoznány maximálně tři vztyčené prsty. Ideální postavení prstů ruky pro stav reprezentující aktivní kreslení je znázorněno na obrázku 3.7a.

Pro srovnání efektivity gesta při kreslení je použita alternativní metoda aktivace kreslení pomocí klávesy `Ctrl`.



(a) (b)

Obrázek 3.7: a) Gesto pro aktivaci kreslení; b) Gesto *ukazatel*.

Zobrazení menu K zobrazení menu dojde jednoduchým statickým gestem, kdy uživatel umístí obě ruce zároveň nad senzor. Gesto by mělo být jednoznačné, jelikož nechtěné vyvolání menu by bylo velice rušivé. Druhá ruka není využita k žádným činnostem, a proto by k nechtěným aktivacím nemělo docházet.

3.4 Potenciální problémy ovládání gesty

V předešlé podkapitole byly zmíněny základní rysy ovládání určitých částí aplikace gesty, neměli bychom ale zapomenout na potenciální problémy, které se mohou objevit. Z problémů senzorů využívajících počítačové vidění zmíněných v podkapitole 2.2 a z vlastních zkušeností při testování senzoru musí být počítáno s překrýváním částí ruky během jejího pohybu. U navrženého gesta pro aktivaci kreslení (podkapitola 3.3) může dojít k naklonění ruky během kreslení (například palcem směrem k senzoru) či překrytí prstu částí ruky a aplikace nemusí rozpoznat statické gesto a dojde tedy k přerušení probíhajícího kreslení.

Dále bude prakticky ověřena schopnost senzoru rozlišit dostatečně přesně polohu prstů a jejich délku. Pokud by se detekovaná délka prstů za běhu programu často měnila, mohlo by to mít za následek například poskakování *ukazatele* prostorem.

Následně by mohla nastat komplikace při pohybu prstů v krajní části reakčního prostoru senzoru, kdy by mohlo dojít ke ztrátě informací o pozici krajního prstu. Tomu by však mělo zamezit omezení detekčního rozsahu senzoru (zmíněným v podkapitole 3.1), kdy aplikace neumožní kurzoru pohybovat se za hranicí rozsahu. O prstech, které jsou sice mimo rozsah

pracovního prostoru, ale stále v detekčním rozsahu, však stále aplikace má přehled, díky „rezervě“, která vznikla tímto omezením rozsahu senzoru.

Pokud z testování vyšla některá další gesta jako nespolehlivá, nabízí se nám možnost jejich nahrazení například stiskem předdefinované klávesy.

Kapitola 4

Implementace

Základem pro implementaci aplikace je grafická sada nástrojů OpenSceneGraph[10] (dále v textu jen jako OSG) distribuovaná pod licencí OpenSource[1]. Jedná se o objektově orientovaný *framework* (tj. ucelená sada nástrojů, knihoven či dalších podpůrných prostředků s cílem zjednodušit implementaci projektů s podobným zaměřením) implementovaný v jazyce `C++` obalující multiplatformní API OpenGL[2] pro tvorbu počítačové grafiky. OSG je celé založeno na tzv. *Scene Graph* konceptu[3], což je základní stromová struktura (graf) použitá k popisu grafické scény. Uzly na nekoncových úrovních reprezentují hierarchii scény a manipulátory se scénou a naopak koncové uzly reprezentují geometrické objekty k vykreslení. OSG podporuje různé nadstavby (angl. *language wrappers*) pro ostatní jazyky mimo `C++` (například `Java`, `C#`, `Python`). Nicméně v tomto případě je použit jazyk `C++`, který přímo podporuje jak OSG, tak i oficiální Leap Motion API ve verzi 1.0.9.8391, jehož základní struktura byla již dříve popsána v podkapitole 2.3.4.

Pro tuto práci nezbytnou komunikaci se senzorem nám výrazně usnadní aktivně vyvíjená knihovna – `osgLeap`[4] – pro OSG aplikace využívající Leap Motion senzor. Jedná se o knihovnu, která poskytuje rozhraní pro základní komunikaci a aktualizaci dat ze senzoru a zároveň implementuje některé základní prostředky pro tvorbu uživatelského rozhraní podporující Leap Motion senzor. Z této knihovny bylo v této práci vycházeno a přímo využíváno jejích součásti. Vývoj aplikace probíhal na operačním systému Windows 8.1 v prostředí Microsoft Visual Studio 2012, ve kterém byla instalována aktuální stabilní verze OSG 3.2.0 z projektu vytvořeného pomocí nástroje CMake[5].

V této kapitole je nejdříve popsána struktura vytvořené aplikace, kdy se zaměříme na implementaci 3D kreslení a funkcionality 2D menu. Na závěr kapitoly následuje popis implementace rozpoznání navržených gest.

4.1 Struktura aplikace

Při implementaci bylo vycházeno z knihovny `osgLeap`, umožňující zobrazení prstů na obrazovce jako 2D ukazatelů, u nichž je také emulován jejich pohyb jako při pohybu kurzoru

[1] Softwarová licence udávající, že software je šířen s otevřeným zdrojovým kódem (viz domovská stránka licence http://opensource.org/).

[2] Viz domovské stránky http://www.opengl.org/.

[3] Stručné informace o SceneGraph konceptu v OSG na http://www.cs.colorado.edu/~kena/classes/5448/s11/presentations/presentationryankroiss.pdf.

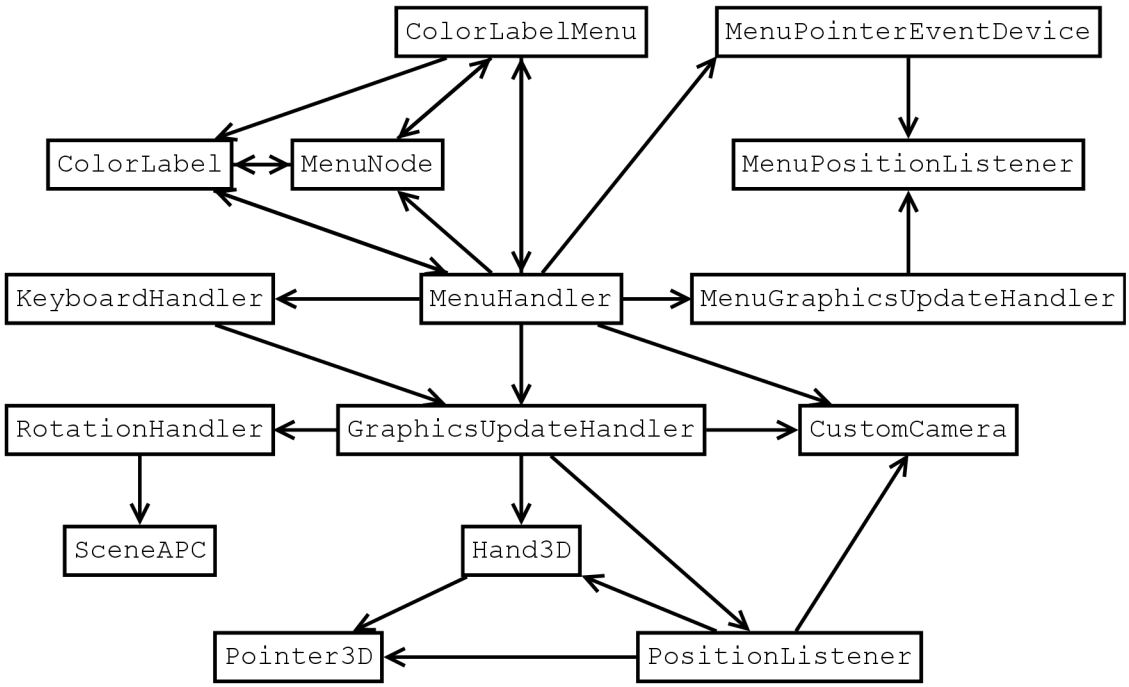
[4] Repozitář a informace o knihovně jsou dostupné na https://code.google.com/p/osgleap/.

[5] Domovská stránka nástroje http://www.cmake.org/.

myši. Díky tomu lze poměrně jednoduše implementovat interakci uživatele s menu. Tato knihovna ale neposkytuje funkcionalitu pro implementaci pohybu ukazatele 3D prostorem, což je pro naši aplikaci zásadní.

Princip řízení běhu aplikace je založen na opětovném zpětném volání při každé aktualizaci hlavního uzlu scény. Reakci na toto zpětné volání zajišťuje třída `MenuHandler`, která je detailněji popsána v podkapitole 4.1.2. Při každé aktualizaci scény je zpětně volán operátor `()`, který prakticky řídí chod celé aplikace. Například vždy předává zpětné volání dále třídě `GraphicsUpdateHandler`, která se na základě informací ze senzoru stará o veškerou funkcionalitu 3D scény. Této třídě se věnuje podkapitola 4.1.1, protože její funkcionalita je zásadní.

Závislosti tříd, které jsou nadále popisovány, znázorňuje diagram na obrázku 4.1. Z něj je patrné, že například třída `MenuHandler` je hierarchicky nad ostatními třídami, a hlavně přímo ovládá třídu `GraphicsUpdateHandler`. Jediné třídy, které mění stavy třídy `MenuHandler`, jsou třídy `ColorLabel` a `ColorLabelMenu`. U nich je to nezbytné, protože se jedná o grafické prvky menu, jež musí nadřazenou třídu upozornit, že došlo k jejich vybrání. Stručný popis funkcionality všech tříd je k nalezení v příloze v tabulce B.1.



Obrázek 4.1: Diagram závislostí tříd

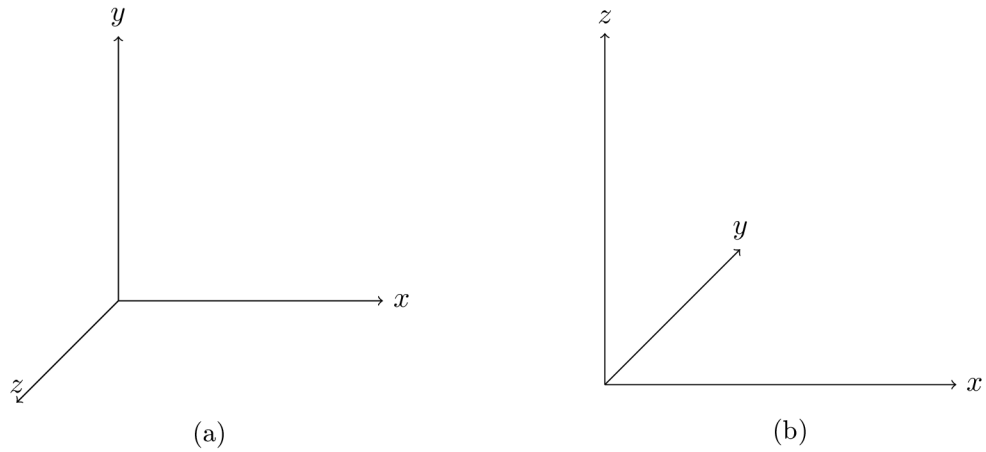
4.1.1 Implementace 3D prostoru a zpracování dat

V této podkapitole je nejprve vysvětleno jakým způsobem jsou získávána data ze senzoru a jak jsou dále interpretována pro použití při 3D kreslení.

Načtení dat ze senzoru Sběr dat ze senzoru a jejich následnou interpretaci v použitelné podobě má na starosti třída `PositionListener`. Při pokynu k aktualizaci dat ze senzoru dojde nejprve k načtení dat ze senzoru z takzvaného snímku. Uchovávány jsou

informace o dvou posledních snímcích ze senzoru, pomocí nichž se provede odstranění neplatných ukazatelů z minulého snímku. Případně se zde nastavuje příznak nalezených gest a dále přiřazení nalezených prstů k rozpoznaným rukám. Pro reprezentaci rukou je použita vlastní třída `Hand3D`, do níž jsou vloženy všechny k ní náležící prsty, reprezentovány naší třídou `Pointer3D`. Vlastních tříd je využito z důvodu zjednodušení operace nad prsty (například rozpoznání aktivního gesta pro kreslení, výběr nejdelšího prstu apod.). U prstů je uchovávána zejména pozice ukazatele v OSG souřadném systému a další informace, například identifikační číslo, délku prstu, apod. Před uložením pozice prstů je také nezbytné přepočítání souřadnic, což je popsáno v následující podkapitole.

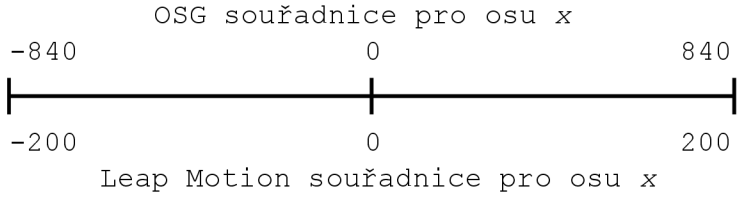
Souřadný systém a pracovní prostor Využitý rozsah senzoru se dynamicky mění v závislosti na použitém rozlišení okna aplikace, jak bylo popsáno v podkapitole 3.1. Pokud by došlo ke změně rozlišení aktivní kamery, přepočítá se a roztáhne operační prostor do předdefinovaných intervalů. Pro každý ukazatel se vždy provádí přepočet souřadnic z Leap Motion souřadného systému do OSG souřadného systému. Leap Motion senzor totiž používá pravotočivou soustavu kartézských souřadnic, jak je znázorněno na obrázku 4.2a. Oproti tomu v prostředí OSG je celý souřadný systém natočen tak, že vektor osy z míří směrem vzhůru a vektor osy y směrem od pozorovatele. Tyto změny jsou znázorněny na obrázku 4.2b.



Obrázek 4.2: a) Souřadný systém u Leap Motion senzoru; b) Souřadný systém u OSG prostředí.

Souřadnice ze senzoru jsou poté „namapovány“ do našeho OSG prostoru jak je pro horizontální rozlišení 1680 pixelů znázorněno na obrázku 4.3. Na něm můžeme pozorovat, že zvolený rozsah $< -200, 200 >$ pro osu x je převeden na rozsah odpovídajícímu aktuálnímu horizontálnímu rozlišení, jen se středem intervalu (určeným velikostí rozlišení) umístěným v nule.

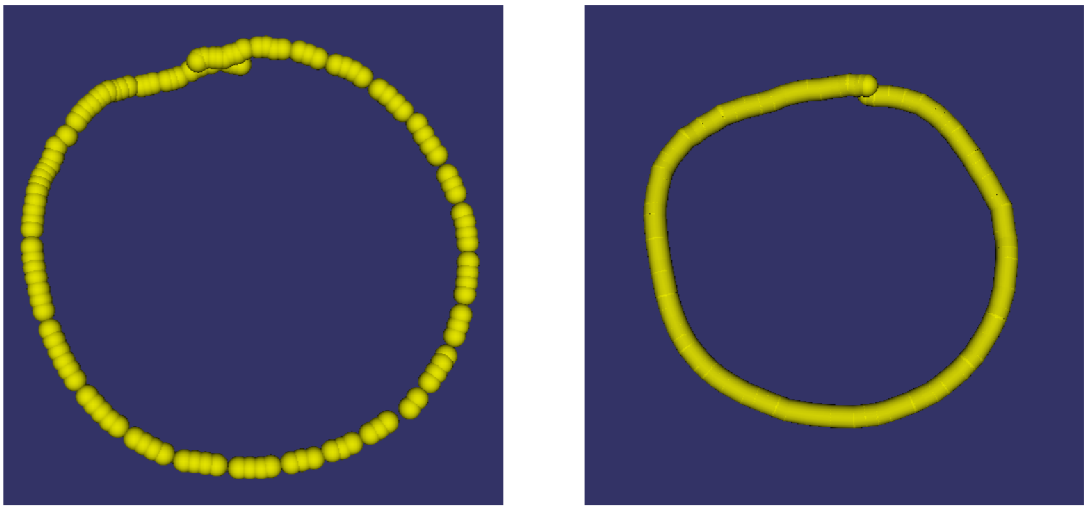
Vyhodnocení dat ze senzoru Veškerou funkcionalitu 3D scény zprostředkovává třída `GraphicsUpdateHandler`, ve které se prakticky řídí celý běh 3D kreslení a s každou aktualizací grafické scény dochází k načtení nových dat ze senzoru. Například při každé změně rozlišení se znovu vytvoří kvádr znázorňující hranice pracovního prostoru, jehož rozměry odpovídají aktuálnímu rozlišení okna aplikace. Následně se zde aktualizuje pozice kamery tak, aby byl vždy viditelný celý pracovní prostor a zároveň co možná nejvíce přiblížený.



Obrázek 4.3: Převedení Leap Motion souřadného systému do OSG

Za běhu programu jsou v této třídě uchovávány stavy nastavené v závislosti dění ve 3D scéně. Například při rozpoznání gesta pro zobrazení menu se okamžitě předá řízení nadřazené třídě pro správu menu. Jinak se provede kontrola rozpoznaných gest, a případně se nastaví příslušný stav odpovídající gestu. Implementace rozpoznání gest je blíže popsána v podkapitole 4.2. Dále se zde vykresluje aktuální pozice *ukazatele* nebo při aktivním kreslení i jeho trajektorie.

Způsoby vykreslování trajektorie Při aktivním kreslení se trajektorie pohybujícího *ukazatele* vykresluje jedním ze dvou následujících způsobů. První z nich při každé aktualizaci scény vykreslí na pozici ukazatele jednoduchý tvar koule. Při rychlejším pohybu však algoritmus nestíhá vykreslovat objekty dostatečně rychle, a tak vznikají v trajektorii znatelné mezery, které jsou patrné z obrázku 4.4a. Druhý způsob, který je znázorněný na obrázku 4.4b, vykresluje mezi dvě poslední zaznamenané pozice *ukazatele* tvar válce s objektem ve tvaru koule umístěným na každém jeho konci. Tento způsob vykresluje o poznání plynulejší trajektorii.



(a) (b)

Obrázek 4.4: a) První způsob vykreslování; b) Druhý spojovaný způsob vykreslování.

Prohlížení scény Pokud dojde k aktivaci režimu *kamery v prstu*, začne se aktivně přesouvat kamera 3D scénou. Tento proces byl původně implementován způsobem popsaným v návrhu v podkapitole 3.2, kdy bylo vycházeno z aktuální pozice *ukazatele* a jeho směrového vektoru do scény. To se ale ihned po realizaci ukázalo jako nepraktické, jelikož docházelo ke „chvění“ ukazatele včetně jeho směrového vektoru. To vedlo k nepříjemným otřesům

kamery, a právě z toho důvodu bylo zvoleno fixování pohledu přesně do středu pracovního prostoru. Směrový vektor kamery je tedy namířen do středu pracovního prostoru a pozice kamery je určena pozicí *ukazatele* vynásobenou konstantou, aby bylo dosaženo lepšího akčního rozsahu kamery.

4.1.2 Implementace 2D menu

Funkcionalitu 2D menu a jeho rozhraní zajišťuje třída `MenuHandler`, která především řídí v předchozích kapitolách zmíněné třídy. Dále je rozebrán postup jeho vytváření a ovládání ostatních tříd touto třídou.

Grafická stránka menu Struktura menu je uložena ve stromové, obousměrně provázané, datové struktuře. Podle této struktury se později vytváří grafická podoba menu. Velikost položek poměrově odpovídá aktivnímu rozlišení, kdy jejich výška je závislá na počtu položek první úrovně a šířka odpovídá pětině horizontálního rozlišení.

Položky v dalších úrovních jsou rovnoměrně roztaženy do celé výšky menu, jak bylo již dříve popsáno v kapitole 3.2. Výběr položky je podle návrhu uskutečněn, pokud ukazatel opustí pravou stranu položky menu. Reakcí na to je vyvolání události běžně volané při kliknutí myši, čímž dojde k výběru položky.

Ovládání aplikace Při aktualizaci scény se vždy nejprve pomocí podřízených tříd načtou data ze senzoru a aktualizuje stav aplikace. Pokud se aplikace dostane do stavu indikujícím zobrazení menu, dojde k zobrazení předem připraveného grafického rozhraní. Následně se při každé aktualizaci kontroluje, jestli nedošlo k vybrání některé z položek menu.

Pro realizaci jsou využity třídy z knihovny `osgLeap`, které se starají o zobrazení 2D ukazatele menu a reakci na jeho pohyb jako na pohyb kurzoru myši. Tyto třídy musely být mírně upraveny, aby bylo možné odstranit přebytečné ukazatele prstů. Knihovna `osgLeap` v základu totiž zobrazuje ukazatele pro veškeré rozpoznané prsty, což způsobovalo nechtěné vyvolání událostí na jejich pohyb. Proto byly tyto třídy upraveny, a vždy je zobrazen právě jeden *ukazatel* menu pro nejdelší rozpoznaný prst.

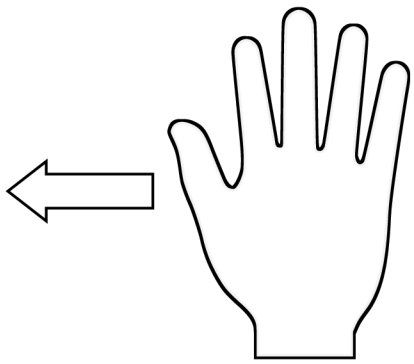
4.2 Implementace rozpoznávání gest

V této kapitole je popsána implementace detekce gest, jež jsou využita v této aplikaci. Jedná se o poměrně jednoduchá gesta, která by měl senzor bez problému rozpoznat.

Dynamické gesto pro otočení scény U tohoto gesta nebyla implementace nijak složitá, jelikož je podporováno přímo senzorem. Konkrétně se jedná o gesto *swipe*, které bylo blíže popsáno v podkapitole 3.3. Při každé aktualizaci hlavního uzlu scény dochází i ke kontrole senzorem rozpoznaných gest. Aby nedocházelo k nechtěným rotacím, gesto není možné provést, pokud je právě aktivní 2D menu nebo aktuálně probíhá kreslení. V neposlední řadě je nezbytné povolení tohoto typu gesta na senzoru a nastavení parametrů jeho aktivace. Praktickým testování byla zvolena pro aktivaci gesta jeho minimální délka `150 mm` a rychlost `1500 mm/s`. Toto nastavení případně bude upraveno na základě výsledků testování.

Dále se z rozpoznaných *swipe* gest vyberou pouze ta, která jsou horizontální. To zjistíme prostým porovnáním x a y složky směrového vektoru gesta. V případě, že je x složka větší, gesto je považováno za horizontální. Určení směru gesta potom probíhá následovně – pokud je x kladné, jedná se o *swipe* gesto vpravo, jinak pokud je x záporné, jedná se o gesto vlevo.

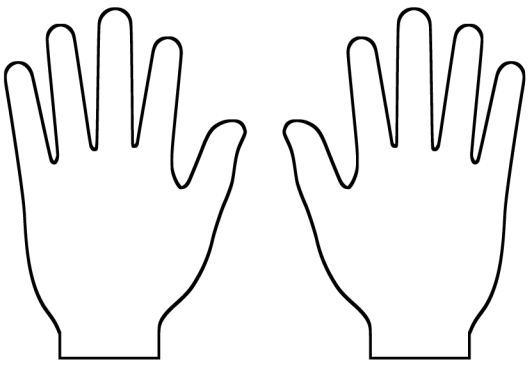
Po implementaci právě popsaného principu však docházelo k nechtěným rotacím scény (například když se při neaktivním kreslení rychleji přesunulo *ukazatelem*). Proto byla dodatečně přidána podmínka určující, že při detekci gesta *swipe* musí být na ruce rozpoznaných všech 5 prstů. Aby bylo tedy toto gesto rozpoznáno, musí být provedeno s otevřenou rukou a prsty dostatečně od sebe. Gesto *swipe* s otevřenou rukou směrem vlevo je znázorněno na obrázku 4.5.



Obrázek 4.5: *Swipe* gesto vlevo s otevřenou rukou

Řízení animace a rotace scény Jelikož prosté skokové natočení scény není příliš uživatelsky přívětivé, rotace uživatelského prostoru je prováděna pomocí animace. Celkem se jedná o 8 různých animací – 4 pro rotaci vlevo a 4 vpravo. Tuto funkcionalitu, včetně uchování stavů o průběhu animací, zajišťuje třída `RotationHandler`. Dále třída `SceneAPC` umožňuje reagovat na konec animace, což není v OSG původně podporováno.

Statické gesto pro zobrazení menu Implementace gesta pro zobrazení menu podle návrhu v podkapitole 3.3 byla původně řešena následovně. Pokud senzor rozpoznal více jak jednu ruku v obraze, okamžitě došlo k přerušení činnosti v 3D prostoru a bylo předáno řízení do nadřazené třídy. Senzor ale často detekoval některé části těla nebo okolní předměty a prohlašoval je za ruce. Proto byla přidána podmínka, že pro aktivaci menu musí být umístěny nad senzor současně obě ruce, každá s pěti nataženými prsty odtaženými od sebe. Toto gesto je znázorněno na obrázku 4.6.



Obrázek 4.6: Statické gesto pro zobrazení menu

Statické gesto pro kreslení Podle návrhu v podkapitole 3.3 bylo rozpoznání gesta pro aktivaci módu kreslení implementováno následovně. Nejprve proběhne kontrola počtu prstů u ruky, kdy musí být na ruce viditelné maximálně 3 prsty. Dále se vybere nejdelší prst zpracovávané ruky a pro každý prst ruky kromě nejdelšího je změřen úhel v radiánech φ svíraný s nejdelším prstem. Původně byl počítán svíraný úhel ve 3D prostoru, nakonec se ukázalo spolehlivější tento úhel počítat pouze v rovině xy. Ten je možné zjistit podle vzorce 4.1 pro výpočet úhlu mezi vektory.

$$\cos(\varphi) = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}| \cdot |\vec{v}|} \tag{4.1}$$

Po upravení předchozí rovnice dostaneme výslednou rovnici 4.2. Hodnota úhlu φ ve stupních je nakonec získána vynásobením zlomkem $\frac{180^\circ}{\pi}$ podle vztahu pro převod radiánů na stupně.

$$\varphi = \arccos\left(\frac{u_1 v_1 + u_2 v_2}{\sqrt{u_1^2 + u_2^2} \cdot \sqrt{v_1^2 + v_2^2}}\right) \tag{4.2}$$

Pokud některý prst svíral s nejdelším prstem úhel větší nebo rovno 30 stupňů, je to považováno za kreslení. Tento princip byl však po jeho realizaci mírně upraven, jelikož senzor občas, byť pouze v jednom snímku, vyhodnotil nesprávně úhel mezi prsty. Právě to vedlo ke chvilkové aktivaci či přerušení kreslení. Z tohoto důvodu byl vytvořen časovač, který zajišťuje prodlevu mezi aktivací nebo deaktivací kreslení. Aby došlo ke změně stavu kreslení, musí aplikace setrvat v novém stavu po celou dobu časového intervalu. Před testováním byla hodnota časového intervalu stanovena na `100 ms` a ověříme ji praktickým testováním. Tento způsob však vede k mírnému zpoždění aktivace či deaktivace kreslení, nicméně subjektivně se toto zpoždění téměř neprojevuje. Naopak tím byly minimalizovány nepříjemné „výpadky“ při vykreslování trajektorie.

Výsledná aplikace Obrázky z prostředí aplikace jsou k dispozici v příloze. Konkrétně se jedná o obrázek C.1 pořízený během výběru položky z menu. Druhý obrázek C.2 znázorňuje aplikaci během kreslení.

Kapitola 5

Testování

V této podkapitole je přímo vycházeno ze souhrnných informací, které uvádí Koji Yatani na svých stránkách[4], jež se mimo jiné zabývají statistickým testováním interakce uživatelů s virtuálním prostředím.

5.1 Návrh testování a postup vyhodnocení

Testování aplikace probíhalo ve dvou krocích na stejné skupině uživatelů (tzv. *within-subject design*). V prvním kroku byla uživatelům předložena podle návrhu zhotovená aplikace, ve které měli uživatelé za úkol vykonat jednoduchý úkol za použití aktivace kreslení gestem. V druhém kroku byli vyzváni ke změně způsobu kreslení na aktivaci kreslení klávesou a provedení téhož úkolu. Pro každý způsob jim byl měřen čas, za který ho vykonali.

Jejich úkol spočíval v nakreslení prvních čtyř písmen abecedy na každou stranu pracovního prostoru tak, aby byla písmena napsána tiskacím písmem a byla čitelná. Uživatel tedy musel nakreslit z prvního pohledu do prostoru písmeno, přerušit kreslení a následně použít gesto pro otočení pracovního prostoru. Tento proces se opakoval dokud nebylo na každé straně pracovního prostoru nakresleno písmeno.

Měření probíhalo až po seznámení uživatelů s rozhraním aplikace a ovládáním gesty. Uživatelé byl vyzváni, aby si vyzkoušeli výběr položek z menu a rozdílné metody aktivace kreslení. Zároveň si mohli vyzkoušet úkol nanečisto. Jakmile se sami považovali za dostatečně obeznámené s rozhraním aplikace, přistoupilo se k provedení úkolu. Po jeho splnění byl uživatel požádán o zpětnou vazbu a připomínky k rozhraní aplikace (viz dotazník v příloze A). Po zpracování výsledků jsou nejčastěji vyskytující se problémy diskutovány a případně opraveny.

Postup vyhodnocení výsledků Výsledky jsou vyhodnoceny pomocí metody *párový t-test*, který slouží k porovnání středních hodnot mezi skupinami. Při tomto testu se vyvrací nulová hypotéza určující, že mezi dobou provádění testovacího úkolu pro obě metody není žádný významný rozdíl. Pokud dojde k jejímu vyvrácení, považujeme za platnou alternativní hypotézu, ze které vyplyne efektivnější (rychlejší) metoda. V našem případě považujeme za nulovou hypotézu tvrzení, že mezi časy při plnění úkolu aktivací gestem a aktivací klávesou není významný rozdíl (neboli střední hodnota časů u první a druhé metody se rovnají).

Výsledkem použití metody *t-test* je pravděpodobnost p. Hodnota p určuje, jaká je pravděpodobnost, že dosáhneme extrémních hodnot při testování, pokud zůstane nulová hypo-

téza v platnosti. Čím menší je tedy hodnota p, tím nepravděpodobnější je udržení nulové hypotézy v platnosti. Hodnota prahu pro hodnotu p, při které považujeme rozdíl za *statisticky významný* je u této metody pevně určena na **0,05**. Stejně tak pokud je hodnota dokonce menší než **0,01**, je rozdíl považován za *statisticky vysoce významný*.

Hodnota p nám nedává přehled o tom, jak moc významný rozdíl mezi jednotlivými testy byl. Proto určíme i koeficient *velikosti efektu* (angl. *effect size*) pomocí Cohenova d, které podle vzorce 5.1 určuje míru efektu mezi porovnávanými metodami. Čitatel $|\mu_1 - \mu_2|$ je rozdíl středních hodnot obou metod a σ je směrodatná odchylka rozdílu hodnot mezi metodami. Tento koeficient určuje velikost rozdílu mezi dvěma průměry ve směrodatných odchylkách.

$$d = \frac{|\mu_1 - \mu_2|}{\sigma} \tag{5.1}$$

Testování bylo prováděno na skupině *patnácti* uživatelů, kteří se pohybují v oboru informačních technologií. Především se jednalo o studenty či absolventy bakalářského a magisterského studia na VUT FIT. Nikdo z nich ale neměl bližší zkušenosti se senzorem tohoto druhu.

5.2 Výsledky testování

Pro výpočet potřebných hodnot pro srovnání metod byl použit statistický program *R*[1].

Aritmetický průměr Nejprve byl vypočítán aritmetický průměr času provádění úkolu pro obě metody. Pro aktivaci gestem vyšel průměrný čas provádění úkolu **27,5** sekund oproti **21,92** sekundám pro metodu *aktivace kreslení klávesou*. *Aktivace kreslení gestem* byla v průměru cca o **5,6** sekund pomalejší. Data nasbíraná v tomto testování jsou dostupná v příloze v tabulce D.1.

Metoda *párový t-test* Metodou *párový t-test* byl zjištěn *statisticky vysoce významný* rozdíl s hodnotou $p < 0,01$ pro dvě rozdílné techniky kreslení, kdy p bylo $3{,}132 \times 10^{-5}$. Velikost efektu určíme pomocí Cohenova d, které vyšlo přibližně **1,55**. Tudíž spadá do horního intervalu, u kterého platí vztah $d >= 0,8$. Velikost efektu je tedy hodnocena jako *velká* (angl. *large size*). Z toho vyplývá, že metoda aktivace klávesou je znatelně rychlejší než metoda aktivace gestem. Postup výpočtu výsledků programem *R* je umístěn v příloze E.1.

Poznatky uživatelů z testování Zde jsou shrnuty některé poznatky z testování aplikace uživateli a z jejich zpětné vazby. Především jsou řešeny problémy, které se objevily během testování.

Nežádoucí kreslení gestem Uživatelé si stěžovali, že při aktivaci kreslení gestem se jim po výběru položky z menu neúmyslně aktivovalo kreslení. Příčinou bylo zřejmě neúmyslné použití gesta pro kreslení i ve výběru menu. Po výběru položky však nestačili včas zaujmout správné postavení ruky. Na základě těchto problémů byl přidán časovač, který po deaktivaci menu po dobu `500 ms` znemožňuje kreslení.

[1] Viz domovské stránky http://www.r-project.org/.

Dále se často podařilo uživatelům aktivovat či deaktivovat kreslení, i když to neměli v úmyslu. Nejvíce se tento nedostatek projevoval, pokud senzor z nějakého důvodu nedokázal rozpoznat pozici prstů přesně. To se často projevovalo v krajních mezích detekčního rozsahu senzoru, kde byla snížená přesnost detekce pozice prstů a tudíž i zkreslený úhel mezi prsty. O tomto nedostatku jsem věděl již před začátkem testování a bohužel se mi ho nepodařilo úplně eliminovat. Celkem znatelný efekt ale měla změna výpočtu úhlu z 3D prostoru pouze v rovině xy. Následně se tento problém vyskytoval, pokud byl senzor osvícen přímým zdrojem světla, čímž byla dramaticky snížena přesnost rozpoznání pozice prstů. S odstraněním zdroje problém vymizel. Nutno podotknout, že senzor na tuto situaci sám upozornil hláškou z kontrolního panelu.

Změna siluety ruky a odlesky V mnoha případech hodinky nebo šperky, které měli uživatelé na rukou zhoršovaly či až znemožňovaly schopnost senzoru rozpoznat správnou pozici prstů nebo dokonce ruky. Zhoršení přesnosti rozpoznání pozice prstů vymizelo po jejich sundání. Tyto předměty zřejmě způsobovaly změnu siluety ruky a senzor tak měl problémy s detekcí. Možnou příčinou by ještě mohly být odlesky od jejich hladkého povrchu.

Rozpoznání gesta rozevřené ruky S gestem pro rotaci pracovního prostoru a zobrazení menu se naskytl problém způsobený tendencí uživatelů držet při tomto gestu prsty u sebe. U těchto gest je stanovena podmínka pěti rozpoznaných prstů pro každou ruku. Senzor bohužel považuje například dva prsty, které se vzájemně dotýkají za jeden. Tento stav bohužel nijak nelze ovlivnit, snažil jsem se ho alespoň kompenzovat snížením požadavku rozpoznaných prstů z pěti na čtyři.

Dále byla po testování snížena rychlost potřebná k vyvolání gesta z `1500 mm/s` na `1200 mm/s`, jelikož se nedařilo uživatelům gesto vyvolat i přes dostatek rozpoznaných prstů.

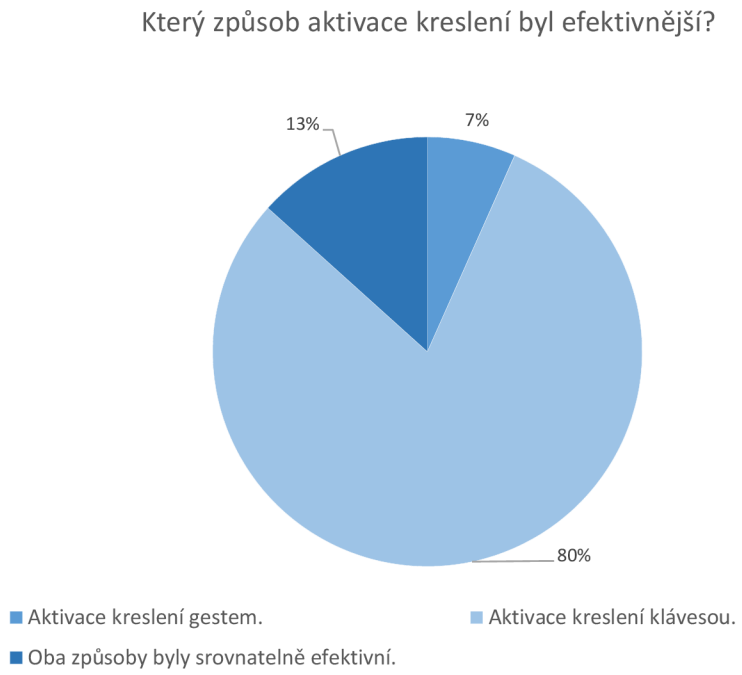
Nesprávný výběr nejdelšího prstu Při testování se objevil problém, kdy byl občas za nejdelší prst byl prohlášen palec, který by měl být kratší. To bylo způsobeno chybou v implementaci, kdy se správně neaktualizovala velikost prstů. Pokud byl palec jednou prohlášen za nejdelší (například druhý prst nebyl ještě natažen), zůstal jako nejdelším dokud nebyl před senzorem skryt. Chybu se částečně podařilo odstranit opravením aktualizace délky prstů. I přesto ho u rukou určitých proporcí senzor občas vyhodnotí jako delší než natažený ukazováček.

Řešení tohoto problému není úplně jednoduché, jelikož nelze jednoduše zjistit, jestli uživatel kreslí levou nebo pravou rukou. Není proto možné jednoznačně identifikovat jednotlivé prsty a zamezit tak vybrání palce jako ukazatele. Řešením by mohlo být umožnění přepínání mezi levorukým a pravorukým režimem.

5.3 Vyhodnocení dotazníku

Jak již bylo řečeno, předmětem testování byly dvě odlišné metody aktivace kreslení, kdy metoda kreslení gestem dopadla ve statistickém vyhodnocení testu hůře. V dotazníku označilo **80 %** respondentů metodu aktivace klávesou jako efektivnější, což je zřejmé z grafu na obrázku 5.1. Jako zdůvodnění, proč vybrali danou odpověď uváděli často uživatelé fakt, že aktivace klávesou je pro ně přirozenější a také, že přesněji umožňuje ovládat průběh kreslení. U aktivace gestem totiž občas docházelo k samovolné aktivaci či deaktivaci kreslení.

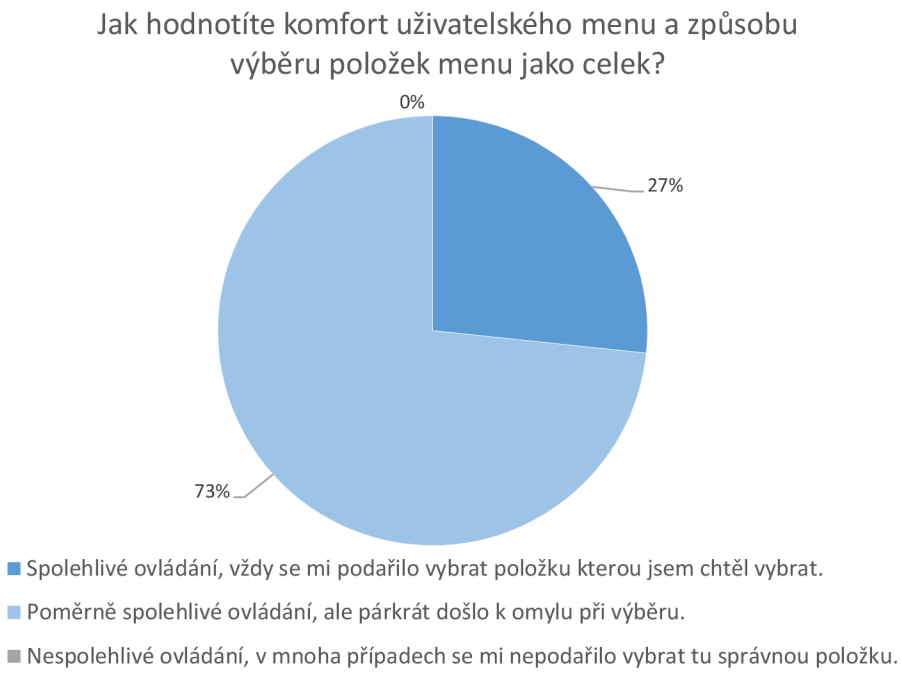
Uživatelé si také stěžovali, že na tento typ ovládání nejsou zvyklí, a tudíž jím chybí potřebná koordinace pohybů. Při testovacím úkolu pro ně byly obtížné přechody mezi gesty – především ukončení kreslení a následná rotace prostorem. uživatelé často omylem naznačili gesto kreslení a docházelo tak k nežádoucímu kreslení.



Obrázek 5.1: Graf s odpovědmi k otázce č.4

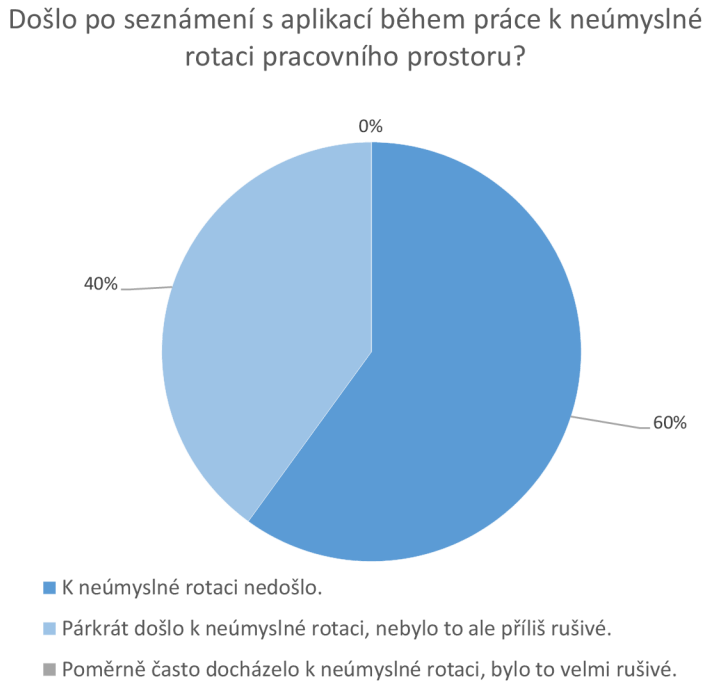
Tato situace by mohla být způsobena tím, že tento způsob ovládání je relativně nový a většina uživatelů s ním ještě neměla zkušenosti. Postoj zaujatý k metodě aktivace gestem pak už jenom prohloubil fakt, že nefungovala zcela spolehlivě ve všech případech.

V dotazníku bylo hodnoceno i rozhraní menu, kdy **27 %** uživatelů se vždy podařilo neomylně vybrat položku z menu. Zbylých **73 %** označilo rozhraní menu za poměrně spolehlivé, ale párkrát došlo k omylu při výběru položky. Výsledky znázorňuje graf na obrázku 5.2.



Obrázek 5.2: Graf s odpovědmi k otázce č.1

Jak je dále patrné z grafu na obrázku 5.3, plných **60 %** respondentů se nesetkalo s neúmyslnou rotací scény. Zbylým **40 %** se tato událost stala jenom výjimečně a nepovažovali to za příliš rušivé.



Obrázek 5.3: Odpovědi k otázce č.2

Na grafu z obrázku 5.4 je znázorněno, že **20 %** respondentům se vždy podařilo otočit pracovním prostorem a dalším **60 %** se to podařilo ve většině případů. Zbylým **20 %** respondentům se často nedařilo toto gesto vyvolat.

Obrázek 5.4: Odpovědi k otázce č.3

Doplňující testování Na skupině *pěti* uživatelů jsem otestoval, zda po delší době (přibližně 10 minut) používání rozhraní a metody aktivace gestem dosahují uživatelé lepších výsledků při plnění stejného úkolu. Uživatelé již byli z předchozího testování seznámeni s rozhraním aplikace a soustředili se tedy spíše na metody kreslení. Data nasbíraná v tomto testování jsou v příloze v tabulce D.2.

Aritmetický průměr časů provádění úkolu byl v tomto případě o poznání nižší – pro metodu gestem **14,6** sekund a **13,62** sekund pro metodu klávesou. Pomocí metody *t-test* **nebyl** zjištěn *statisticky významný rozdíl*, jelikož hodnota p byla $0,066$. Velikost efektu – Cohenovo d – vyšlo v tomto případě **1,12** a tento efekt opět můžeme prohlásit za *velký*. Výpočet výsledků programem *R* je umístěn v příloze E.2.

Na výsledcích doplňujícího testování lze pozorovat znatelné zlepšení metody aktivace gestem. Oproti předchozímu testování vyšlo totiž Cohenovo d o **0,43** menší, což naznačuje snižování rozdílu mezi metodami při jejich delším používání. Jelikož nebyla sledována kvalita uživatelem nakreslených písmen, může být tento výsledek značně zavádějící. Čas provádění u obou metod totiž klesl téměř na polovinu původního času což by mohlo naznačovat klesající kvalitu vytvořených písmen. Pro potvrzení skutečné příčiny tak znatelného zlepšení by bylo třeba rozsáhlejšího testování.

Nutno podotknout, že během obou testování nebyla uznána písmena, která nebyla čitelná a test se v tomto případě opakoval. Zajímavé by mohlo být srovnání kvality písmen mezi jednotlivými uživateli vzhledem k času jejich vytváření. Bohužel výsledky kreslení nebyly zaznamenávány a nemohou být z tohoto hlediska porovnávány.

Známé problémy Na závěr této podkapitoly jsou zmíněny známé problémy aplikace, které se vyskytly během testování. Většina z nich nebyla cílem této práce nebo již nestihla být vyřešena. Aby nebyly tyto problémy jenom přehlíženy, jsou zde diskutovány a případně je navrženo jejich řešení.

Výkonnostní problémy Jelikož třída `osg::ShapeDrawable` použitá k uchování vykreslované trajektorie ukazatele není navržena k tomu, aby se ve scéně vyskytovala v řádů stovek či tisíců, způsobuje to při značném zaplnění scény výkonnostní problémy. Protože je tato aplikace určena spíše k testovacím účelům, není tento problém příliš podstatný. Částečně byl tento problém řešen snížením detailů u vykreslovaných tvarů na polovinu. Toto řešení sice znatelně zlepšilo celkovou plynulost aplikace, určitě ho ale nelze považovat za konečné.

Dále jsem zkoumal řešení tohoto problému pomocí metody *level of detail*, která snižuje počet vykreslených polygonů u objektů v závislosti na vzdálenosti kamery. Od tohoto řešení však nakonec bylo upuštěno, jelikož se kamera většinou nachází ve fixní vzdálenosti a výsledky použití této metody nebyly příliš znatelné.

Lepším řešením tohoto problému by například mohlo být vymodelování vykreslovaných tvarů v některém z grafických programů a jejich následné umisťování těchto modelů na pozici ukazatele. To by mohlo vést k podstatně nižším požadavkům na výkon.

Přizpůsobení velikosti menu V závěru jsem již nestíhal vyřešit problém, který se vyskytuje, pokud dojde za běhu aplikace ke změně rozlišení. Rozhraní menu se totiž aktivně nepřizpůsobuje velikosti okna a dochází tak k jeho deformaci a při extrémním zmenšení okna až k nepoužitelnosti.

5.4 Shrnutí výsledků testování

Celkově byli uživatelé spokojeni s rozhraním menu, kdy se v něm poměrně dobře orientovali a úspěšně se jim dařilo vybírat jeho položky. Obdobně úspěšné bylo i ovládání rotace scény gesty, téměř nedocházelo k neúmyslným rotacím a 80 % uživatelů se ve většině případů podařilo bez problému otočit pracovním prostorem. Menší potíže ze začátku činilo uživatelům vytvoření gesta rozevřené ruky, kdy se museli obzvlášť soustředit na prsty, aby byly všechny v dostatečné vzdálenosti a senzor je tak bez problému rozpoznal.

Hlavním cílem testování bylo **porovnání metody aktivace kreslení gestem s metodou aktivace kreslení klávesou**. Bohužel se aktivace gestem nemohla rovnat přesnosti a spolehlivosti stisku klávesy. Pro uživatele bylo celkem obtížné zkoordinovat přechody mezi gesty tak, aby neaktivovali některé další gesto neúmyslně. Z výsledků je však patrné zlepšení v interakci gesty u uživatelů, kteří s aplikací déle pracovali oproti čerstvě „zaučeným“ uživatelům. To by mohlo naznačovat, že prvotní negativní reakce na interakci s aplikací čistě pohybem rukou by mohly být překonány a v budoucnu by mohly aktivně doplňovat již zažité metody interakce. Pro potvrzení těchto předpokladů by však bylo nutné rozsáhlejší testování vlivu zkušeností uživatelů na rozdíl mezi metodami aktivace kreslení.

Kapitola 6

Závěr

Hlavním cílem práce bylo podle zadání navrhnout a vytvořit aplikaci pro 3D kreslení, kterou bude možné ovládat čistě za pomocí gest. Vzniklá aplikace je určena hlavně ke srovnání a zhodnocení kvality ovládání uživatelského rozhraní gesty. V této práci jsem se snažil nastínit základní problematiku sledování lidského těla, a to především pomocí počítačového vidění. Byly zmíněny problémy, které s rozpoznáváním gest úzce souvisejí a bylo nastíněno i jejich řešení.

Při vzájemném srovnání dnes běžně dostupných senzorů pro sledování lidského těla jsem podrobněji zkoumal senzor Leap Motion.

Další důležitou součástí práce byl samotný vývoj aplikace pro 3D kreslení, kdy jsem nejprve zpracoval její návrh. Ten se skládal z popisu uživatelského rozhraní obsahující návrh gest pro běžně prováděné úkony a návrh ovládání menu čistě pomocí senzoru.

Aplikaci jsem následně úspěšně implementoval a tento proces popsal v implementační části. Tam jsem kromě implementačních detailů popsal i problémy řešené během realizace.

V závěru práce jsem provedl testování vytvořené aplikace na skupině uživatelů a následné vyhodnocení. V testování byla především hodnocena efektivita metody aktivace kreslení gestem oproti aktivaci klávesou. Z něj vyšla lépe metoda aktivace klávesou především proto, že se u druhé metody uživatelům občas podařilo nechtěně aktivovat kreslení. Druhé testování na užší skupině uživatelů však potvrzuje citelné zlepšení při metodě aktivace kreslení gestem. Pokud uživatelé s aplikací touto metodou manipulovali delší dobu, přiblížili se efektivitou (časem provádění úkonu) metodě s klávesou.

Vytvořená aplikace slouží především k demonstračním účelům pro manipulaci gesty a kreslení ve 3D prostoru. Za zajímavé rozšíření funkcionality považuji přidání možnosti kreslit různé geometrické útvary (například kvádr, úsečku) a dále s nimi manipulovat.

Zajímavou alternativou, kde by aplikace mohla mít i praktičtější využití, by mohlo být vytvoření mřížky ve 3D prostoru, ke které by se aktivně „přichytávaly“ vytvářené geometrické objekty. Mřížka by umožňovala přesné rozmístění objektů po scéně a pravděpodobně by i usnadnila orientaci v 3D prostoru.

Literatura

[1] Bowman, D.; Kruijff, E.; LaViola, J.; aj.: *3D User Interfaces: Theory and Practice*. Pearson Education, 2004, ISBN 0-201-75867-9.

[2] Hummels, C.; Stappers, P.: Meaningful gestures for human computer interaction: beyond hand postures. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, Apr 1998, s. 591–596, doi:10.1109/AFGR.1998.671012.

[3] Khoshelham, K.; Elberink, S. O.: Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. *Sensors*, ročník 12, č. 12, 2012: s. 1437–1454.
URL http://www.mdpi.com/1424-8220/12/2/1437/

[4] Koji Yatani: Statistics for HCI Research. [online], 2014, citováno 18. dubna 2014.
URL http://yatani.jp/HCIstats/HomePage

[5] Leap Motion, Inc: Leap Motion Controller. [online], 2013, citováno 26. listopadu 2013.
URL https://developer.leapmotion.com/documentation

[6] Leap Motion Team: Leap Motion. [online], 2013, citováno 7. března 2014.
URL https://www.leapmotion.com/blog/rethinking-menu-design-in-the-natural-interface-wild-west/

[7] McNeill, D.: *Hand and mind*. University of Chicago Press, 1992, ISBN 02-265-6132-1.

[8] Microsoft Corporation: Kinect for Windows. [online], 2013, citováno 24. listopadu 2013.
URL https://www.microsoft.com/en-us/kinectforwindows

[9] Nintendo of America, Inc.: Nintendo Wii. [online], 2013, citováno 25. listopadu 2013.
URL http://www.nintendo.com/wii

[10] OSG Community: OpenSceneGraph. [online], 2014, citováno 13. února 2014.
URL http://trac.openscenegraph.org/projects/osg/

[11] Sony Computer Entertainment America LLC: PlayStation Move. [online], 2013, citováno 24. listopadu 2013.
URL http://us.playstation.com/ps3/playstation-move

[12] Sony Computer Entertainment Inc.: PlayStation Move. [online], 2014, citováno 6. ledna 2014.
URL http://scei.co.jp/corporate/release/100311e.html

[13] Turk, M.: Gesture Recognition. In *Handbook of virtual environments*, editace K. M. Stanney, Lawrence Erlbaum Associates, 2002, ISBN 08-058-3270-X, s. 211–223.

[14] Weichert, F.; Bachmann, D.; Rudak, B.; aj.: Analysis of the Accuracy and Robustness of the Leap Motion Controller. *Sensors*, ročník 13, č. 5, 2013: s. 6380–6393.
URL http://www.mdpi.com/1424-8220/13/5/6380/

Příloha A

Dotazník

1. Jak hodnotíte komfort uživatelského menu a způsobu výběru položek menu jako celek?
   (a) Spolehlivé ovládání, vždy se mi podařilo vybrat položku kterou jsem chtěl vybrat.
   (b) Poměrně spolehlivé ovládání, ale párkrát došlo k omylu při výběru.
   (c) Nespolehlivé ovládání, v mnoha případech se mi nepodařilo vybrat tu správnou položku.
2. Došlo po seznámení s aplikací během práce k neúmyslné rotaci pracovního prostoru?
   (a) K neúmyslné rotaci nedošlo.
   (b) Párkrát došlo k neúmyslné rotaci, nebylo to ale příliš rušivé.
   (c) Poměrně často docházelo k neúmyslné rotaci, bylo to velmi rušivé.
3. Podařilo se vám vždy otočit pracovním prostorem aplikace?
   (a) Ano, vždy se mi podařilo vyvolat gesto pro otočení pracovního prostoru.
   (b) Většinou se mi podařilo otočit pracovním prostorem.
   (c) Často se mi nepodařilo vyvolat gesto pro otočení prostorem.
4. Který způsob aktivace kreslení byl efektivnější?
   (a) Aktivace kreslení gestem.
   (b) Aktivace kreslení klávesou.
   (c) Oba způsoby byly srovnatelně efektivní.
5. Prosím odůvodněte výběr předchozí možnosti.
   (a) .....
6. Vaše připomínky a návrhy k aplikaci
   (a) .....

Příloha B

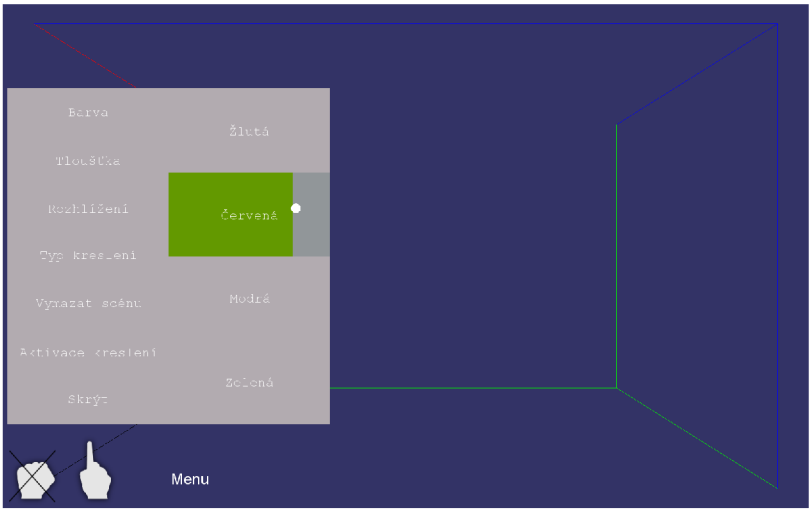
Přehled tříd

Název třídy	Stručný popis
`PositionListener`	Provádí načtení a zpracování dat ze senzoru.
`GraphicsUpdateHandler`	Zajišťuje kompletní funkcionalitu 3D kreslení a uchovává stavy aplikace.
`RotationHandler`	Provádí animace rotace scény a uchovává stavové informace o jejich průběhu.
`SceneAPC`	Umožňuje informovat nadřazenou třídu o konci animace.
`Hand3D`	Třída obsahující informace o načtené ruce a umožňuje provádět operace nad prsty.
`Pointer3D`	Ukazatel ve 3D prostoru reprezentující prst.
`MenuHandler`	Řídí běh celého programu, ovládá podřazené třídy a poskytuje funkcionalitu menu.
`KeyboardHandler`	Umožňuje reagovat na stisknutí kláves a informuje o tom třídě pro správu 3D prostoru.
`MenuNode`	Jedna položka obousměrně vázaného stromu uchovávající strukturu menu.
`ColorLabelMenu`	Grafické znázornění první úrovně menu.
`ColorLabel`	Grafické znázornění druhé úrovně menu, poskytující alternativní možnost výběru položky menu.
`MenuPositionListener`	Slouží k získání informace o poloze 2D ukazatele a vybírá nejdelší prst.
`MenuPointerEventDevice`	Emuluje pro 2D ukazatel události stejně jako pro pohyb myši.
`MenuGraphicsUpdateHandler`	Pohybuje grafickým objektem znázorňující 2D ukazatel.

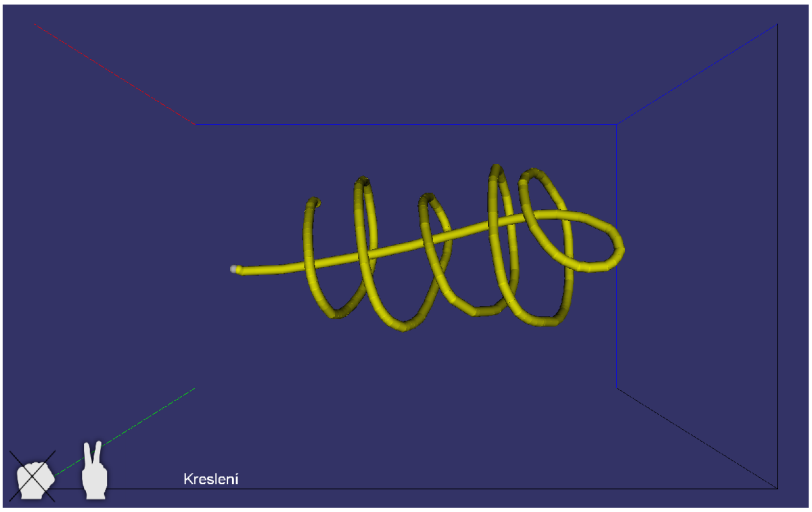
Tabulka B.1: Stručný popis činnosti tříd

Příloha C

Obrázky aplikace



Obrázek C.1: Obrázek aplikace se zobrazeným menu



Obrázek C.2: Obrázek aplikace během kreslení

Příloha D

Tabulky z testování

Pořadí	**Čas provádění gestem [s]**	**Čas provádění klávesou [s]**
1	21,49	14,33
2	30,48	25,42
3	33,6	23,18
4	26,14	20,78
5	25,97	25,41
6	32	23,72
7	30,4	27,6
8	23,54	23,64
9	29,74	18,87
10	21,78	15,45
11	27,04	22,01
12	33,2	24,4
13	30,64	21,5
14	22,48	19,6
15	23,92	22,99

Tabulka D.1: Nasbíraná data z prvního testování

Pořadí	**Čas provádění gestem [s]**	**Čas provádění klávesou [s]**
1	17,68	16,43
2	12,34	12,87
3	14,33	13,1
4	13,12	11,91
5	15,53	13,78

Tabulka D.2: Nasbíraná data z druhého testování

Příloha E

Výpočty výsledků testování

```
# Hodnoty k výpočtům
value <- c(21.488,30.48,33.6,26.14,25.97,32,30.4,23.544,29.74,21.78,
           27.04,33.2,30.64,22.48,23.92,14.33,25.42,23.18,20.78,25.41,
           23.72,27.6,23.64,18.87,15.45,22.01,24.4,21.5,19.6,22.99)
group <- c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1)
data <- data.frame(group, value)

# Výpočet metodou t-test
t.test(data[data["group"] == 0, 2], data[data['group'] == 1, 2], paired=T)

        Paired t-test

data:  data[data["group"] == 0, 2] and data[data["group"] == 1, 2]
t = 6.0223, df = 14, p-value = 3.132e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 3.585194 7.551340
sample estimates:
mean of the differences
              5.568267

# Výpočet Cohenova d
5.568267 / sd(data[data$group == 0, 2] - data[data$group == 1, 2])
[1] 1.554962
```

Listing E.1: Postup výpočtu při prvním testování

```
# Hodnoty k výpočtům
value <- c(17.68,12.34,14.33,13.12,15.53,16.43,12.87,13.1,11.91,13.78)
group <- c(0,0,0,0,0,1,1,1,1,1)
data <- data.frame(group, value)
# Výpočet metodou t-test
t.test(data[data["group"] == 0,2], data[data['group'] == 1,2], paired=T)

        Paired t-test

data:  data[data["group"] == 0, 2] and data[data["group"] == 1, 2]
t = 2.51, df = 4, p-value = 0.06606
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.1042396  2.0682396
sample estimates:
mean of the differences
                 0.982

# Výpočet Cohenova d
0.982 / sd(data[data$group == 0, 2] - data[data$group == 1, 2])
[1] 1.122509
```

Listing E.2: Postup výpočtu při druhém testování