F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Diplomová práce

Rozhraní rozšířené reality pro více zařízení

Bc. Jan Hamet
Otevřená informatika - Interakce počítače s uživatelem

Září 2022, květen 2023
Vedoucí práce: MgA. Vojtěch Leischner

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hamet** Jméno: **Jan** Osobní číslo: **483735**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Specializace: **Interakce člověka s počítačem**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Rozhraní rozšířené reality pro více zařízení

Název diplomové práce anglicky:

Multi device augmented-reality Interface

Pokyny pro vypracování:

Cílem práce je vyvinout a ověřit použitelnost rozhraní rozšířené reality pro více zařízení. Kombinujeme videoprojekci na skleněný válec z několika videoprojektorů s aktivním LED displejem umístěným za válcem. Cílem je zkombinovat více aktivních zobrazovacích povrchů za účelem vytvoření jednotného rozhraní pro uživatele uvnitř skleněného válce. Systém reaguje na uživatele tím, že pomocí kamery snímá jeho gesta a hlas.
Úkoly:
1. Vyviňte systém pro videoprojekci sestávající z několika video projektorů a monitoru. Pro videoprojekci využijte techniky pro video mapping tak, aby se zobrazený obsah jevil pro uživatele nezdeformovaný zakřiveným povrchem válce.
2. Synchronizujte obraz z více video projektorů a více počítačů rendrujících obraz.
3. Vyviňte grafické uživatelské rozhraní, které umí zobrazit text, obraz, video a ovládací prvky a reaguje na uživatele.
4. Implementujte interaktivitu pomocí webkamery a změřte přesnost snímání gest.
5. Změřte jak se mění pozornost uživatelů mezi různými vrstvami rozhraní.
6. Otestujte systém na alespoň pěti uživatelích a ověřte polostrukturovanými rozhovory jejich uživatelskou zkušenost.
Vyvoďte závěry pro další výzkum.
Student má k dispozici: 4* video projektor, monitor / TV, skleněný válec, webkamera, centrální PC, 4* mini PC.
Návrh postupu:
Nejprve vyřešte deformaci projekčního povrchu. Vzhledem k topologii válce je možné využít manuální kalibraci. Tuto funkci nabízí knihovny pro Processing Java (https://processing.org/) nebo Openframeworks (https://openframeworks.cc/). Změřte výkon frameworku metrikou vyrendrovaných framů za sekundu. Následně ve vybraném frameworku vytvořte objektově programované komponenty grafického rozhraní. Při návrhu mějte na zřeteli, že volání komponent bude probíhat decentralizovaně na N počítačích propojených v lokální síti. Centrální počítač vysílá instrukce a komunikuje s periferními počítači. Každý periferní počítač rendruje obraz pro jeden projektor. Díky tomu dosáhneme škálovatelnosti pro více projektorů. Nakonec implementujte trekování gest kamerou (případně i pohledu/očí uživatele) a také hlasové příkazy.

Seznam doporučené literatury:

Jakobsen, M. R., Jansen, Y., Boring, S., & Hornbæk, K. (2015). Should I stay or should I go? Selecting between touch and mid-air gestures for large-display interaction. Human-Computer Interaction--INTERACT 2015: 15th IFIP TC 13 International Conference, Bamberg, Germany, September 14-18, 2015, Proceedings, Part III 15, 455–473. Springer.
Wittorf, M. L., & Jakobsen, M. R. (2016). Eliciting Mid-Air Gestures for Wall-Display Interaction. Proceedings of the 9th Nordic Conference on Human-Computer Interaction. Presented at the Gothenburg, Sweden. doi:10.1145/2971485.2971503
Oshima, K., Moser, K. R., Rompapas, D. C., Swan, J. E., Ikeda, S., Yamamoto, G., ... Kato, H. (2016). SharpView: Improved clarity of defocused content on optical see-through head-mounted displays. 3DUI 2016,173–181.
Maynes-Aminzade, D., Pausch, R., & Seitz, S. (2002). Techniques for interactive audience participation. Proceedings. Fourth IEEE International Conference on Multimodal Interfaces, 15–20. IEEE.
Ishii, H., & Ullmer, B. (1997). Tangible bits: Towards seamless interfaces between people, bits and atoms. Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, 234–241.
Flusser, V, (2013). Za filosofii fotografie. Fra.
Jordà, S., Geiger, G., Alonso, M., & Kaltenbrunner, M. (2007). The reacTable: Exploring the synergy between live music performance and tabletop tangible interfaces. Proceedings of the 1st Int. Conference on Tangible and Embedded Interaction, 139–146.
McLuhan, M., & Fiore, Q. (2001). The medium is the massage. Gingko Pr.

Jméno a pracoviště vedoucí(ho) diplomové práce:

MgA. Vojtěch Leischner katedra softwarového inženýrství FIT

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **02.03.2023** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **16.02.2025**

[redacted] [redacted] [redacted]

MgA. Vojtěch Leischner
podpis vedoucí(ho) práce

podpis vedoucí(ho) ústavu/katedry

prof. Mgr. Petr Páta, Ph.D.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

25.4. 2023
Datum převzetí zadání

[redacted]
Podpis studenta

/ Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 25. 5. 2023

.....

Abstrakt / Abstract

Cílem práce je vytvořit uživatelské rozhraní pro rozšířenou realitu na velkoformátovém průhledném skleněném válci. Uživatel stojí uprostřed válce, na který je promítán obsah z několika videoprojektorů a zároveň může pozorovat scénu za sklem. Rozhraní je multimodální a umožnuje interakci pomocí hlasu a gest.

Rozebíráme různé přístupy k řešení návrhu rozhraní a následně i zvolenou variantu. V rámci zvolené varianty popisujeme hardwarovou přípravu prototypu pro projekci na velkoformátový cylindrický displej a implementaci aplikace řídící synchronizované zobrazování celistvého obrazu složeného z více projektorů.

Software sestává ze samostatné aplikace pro video mapping na každý projektor, jedné řídící aplikace, která zajišťuje, že více projektorů tvoří dohromady jednolitý povrch pro interakci pomocí gest a hlasu a z interakčních aplikací starajících se o rozpoznávání a zpracování speciální interakce.

Na závěr popisujeme uživatelské testování implementovaného rozhraní se zaměřením na sledování jejich pozornosti mezi rozhraním na průhledném displeji a reálnou scénou za ním, společně s celkovým testováním použitelnosti a uživatelské přívětivosti multimodální interakce.

Klíčová slova: uživatelské rozhraní pro rozšířenou realitu; rozšířená realita; víceprojektorová projekce; skládaná projekce; uživatelské rozhraní na průhledném displeji; vícenásobná projekce; projekce na sklo; projekce na válec.

We have created and tested the augmented reality user interface for a transparent large-format cylindrical display. The interface includes hand gestures and voice control.

We present various approaches for designing the user interface. We discuss the hardware setup of a cylindrical display prototype and the distributed software for the multiple synchronized video projectors. Furthermore, we demonstrate a standalone client application for video mapping onto the cylindrical display, together with master software that ensures that multiple clients act as a single surface for seamless interaction via gestures and voice.

Finally, we discuss the user testing of the interface with a focus on attention tracking between the GUI on the transparent display and the natural scene behind it, along with overall usability and user-friendliness testing of the multimodal interaction.

Keywords: augmented reality interface; opticalSeeThrough display; multiDevice projection; projection on the glass; hand gesture controlled interface; optical SeeThrough UI; optical SeeThrough user interface; cilindrical projection system.

Title translation: Multidevice augmented reality Interface

Obsah /

Kapitola 1
Úvod

V posledních letech dochází ke značnému rozvoji rozšířené reality (anglicky augmented-reality, odtud zkratka AR), která umožňuje jako doplněk k reálné scéně, jež můžeme vidět vlastníma očima, zobrazovat dodatečný virtuální obsah. Nejvíce se v tomto odvětví posouvají technologie využívající mobilní telefony a jejich fotoaparát, či stále se rozvíjející AR brýle. Obě tyto technologie ale vyžadují stálou přítomnost zařízení v ruce, či nasazeného na hlavě.

Tato práce oproti výše zmíněným přístupům cílí na technologii využívající mnohem většího, a navíc průhledného displeje. Ne každý doma asi využije okno, které by na svém povrchu dokázalo zobrazit nějaký dodatečný informační obsah o vnějším okolí, ale jistě existují i místa, kde je rozhled a představa o vnějším dění důležitým faktorem, jenž lze pomocí cílených pomůcek jen posílit. Konkrétně se diplomová práce, u této zatím spíše stále sci-fi technologie, zaměřuje na uživatelské rozhraní, které by průhledné displeje mohly využívat a vnímání takového rozhraní uživatelem. Na reálném příkladu si to tak lze představit například u řídící místnosti tovární haly, vodní přehrady či letištní řídící věže, kde by pracovník mohl díky pohledu z proskleného okna udržovat všeobecný přehled o dění za sklem a při tom si nechat na okně zobrazovat doplňující informace o jednotlivých letadlech, částech přehrady či výrobních linkách.

Je otázkou, zda tento netradiční přístup k zobrazování digitálního obsahu může soutěžit cenou či náročností realizace v porovnání s kamerou a více monitory. I přesto však můžeme najít odvětví, kde nalezne své budoucí trvalé uplatnění. Jeho předností je, že není třeba mít žádné zařízení přímo na těle a umožňuje nezkreslený a nerušený výhled na okolí.

1.1 Analýza problému

Cílem práce je vyvinutí aplikace s uživatelským rozhraním zobrazeným na velkoformátovém průhledném displeji, design hardwaru a software pro projekci na velkoformátový displej a sledování uživatelovy pozornosti během práce za účelem validace rozhraní.

Velkoformátový displej je skleněný průhledný válec o průměru 2,5 m a výšce cca 3 m, skládající se ze šesti samostatných skleněných plátů, jež lze vůči sobě posouvat. Uživatel stojí uprostřed válce, odkud sleduje obsah promítaný na jeho vnitřní stěnu. Fotografii místnosti s válcem, kde se vše odehrává, lze vidět na obrázku 1.1.

Hlavním požadavkem na aplikaci je tedy vytvoření uživatelského rozhraní, které bude uživateli zobrazovat dodatečný obsah a informace podporující to, co lze vidět za tímto displejem. Aplikace by zároveň měla umožňovat základní uživatelské interakce se zobrazovaným uživatelským rozhraním, jako je navigace po obrazovce a posun objektů. Technicky musí celá implementace umožňovat také promítnutí tohoto rozhraní na zakřivenou stěnu válce, a to ideálně po celém jeho povrchu.

Druhým požadavkem, následujícím po úspěšné implementaci výše zmíněného rozhraní, je provedení uživatelských testů s alespoň pěti uživateli. Tyto testy se zaměřují

Obrázek 1.1. Fotografie místnosti s válcem ze dvou úhlů.

na vnímání rozhraní uživatelem a sledování jeho pozornosti mezi reálnou scénou a projekcí, použitelnost a uživatelskou přívětivost ovládání pomocí gest a hlasu. Testování také zahrnuje polostrukturovaný rozhovor, jež testuje uživatelský zážitek z používání aplikace.

1.2 Návrh designu

Na začátek je dobré se podívat po již existujících řešeních, zaobírajících se podobným problémem. Ve spojitosti s takzvanými optical see-through průhlednými displeji jsou v současnosti všeobecně známy a vyvíjeny především různé druhy brýlí, případně čoček, pro rozšířenou realitu. Tento druh technologie se ale zaměřuje na umístění displeje přímo před oko uživatele a bývá díky tomu fixně spojen s jeho hlavou. Nás ale více zajímají displeje, jež nejsou nijak svázány s uživatelem a umožňují během nerušeného rozhledu skrz displej do okolí za ním zobrazovat dodatečné informace na jeho povrchu. Tomuto se z existujících (již komerčních) řešení přibližují například head-up displeje (zkratka HUD [1]) v některých moderních autech nebo letadlech.

Myšlenkou HUD displejů je zobrazovat pro řidiče či pilota důležité informace přímo v jeho zorném poli, a přitom mu nijak nebránit ve výhledu z auta. Jednodušší verze těchto displejů většinou zobrazují pouze základní informace jako aktuální rychlost, otáčky motoru či stav nádrže, které jsou jinak běžně viditelné na palubní desce za volantem. Pokročilejší verze těchto displejů však zapojují již i prvky rozšířené reality a snaží se přes část předního skla zobrazovat například i navigaci, varování před srážkou při nedodržení bezpečné vzdálenosti apod. Co se týče technologie displeje, bývají využívány projekce na vlastní průhlednou skleněnou desku či přímo na povrch předního skla.

Z automobilových HUD displejů si ale díky jejich současné velikosti nelze příliš mnoho převzít do našeho řešení, a tak je třeba řešit tento koncept samostatně.

Lze si představit a samostatně uvažovat dvě základní varianty, z nichž první je kombinace průhledného displeje s dalším umělým displejem za ním a druhá je stejná jako již v úvodu zmiňované příklady, tedy průhledný displej s reálnou scénou za ním.

U první varianty lze předpokládat, že by oba displeje měli dokázat vzájemně komunikovat a dokázat spolu synchronizovat zobrazovaný obsah včetně určitého typu uživatelského rozhraní. V realitě si takový případ pak dokáži představit u simulačních komor, kde by se na vnějším displeji zobrazovala hlavní část dat, či simulace reálné scény, a na průhledném displeji pak doplňkové informace včetně rozhraní pro ovládání a další

Obrázek 1.2. Ukázka HUD displejů od firmy Hudway (převzato z [2]).

uživatelský vstup. Otázkou pak ale zůstává, jak velký benefit v uživatelské přívětivosti by přineslo takovéto oddělení dvou vrstev oproti jednoduché integraci obou vrstev do jednoho obrazu a displeje. Určitým benefitem pro oddělené displeje by pro uživatele mohlo být myšlenkově snazší oddělení obou vrstev a jednodušší přenos pozornosti či soustředění mezi nimi.

Za předpokladu že hlavním účelem průhledného displeje má být doplňování a rozšiřování informací o tom, co se děje za ním, působí zmíněný benefit naopak kontraproduktivně, jelikož bychom pravděpodobně chtěli, aby kombinace průhledného displeje a reálné scény byla pro uživatele jednotná a aby se v ideálním případě omezilo nutné střídání pozornosti na minimum. Na toto cílíme především u druhé z variant, kterou je kombinace průhledného displeje s reálnou scénou.

Zde si lze reálné užití představit v několika případech. Prvním z nich může být již zmíněný příklad kontrolní řídící místnosti na letišti, či ve výrobní hale, kde operátor sleduje dění za proskleným oknem a na něm se mu mohou zobrazovat informace o konkrétních letadlech na letištní ploše, stavu nákladu atd. Velkou výhodou by zde také bylo použití funkcí rozšířené reality pro pozicování informací přímo k objektům, k nimž náleží, a celková dynamika rozhraní ovlivňovaná změnami ve venkovním prostředí. Další užití si lze představit u osvětlovačů či operátorů speciálních efektů na větších koncertech a divadelních představeních. Zde bude rozhled a přehled o aktuální scéně důležitým faktorem a ovládací rozhraní umístěné na průhledném displeji před touto scénou by mohlo být použitelným příkladem. Poslední, a asi dle mého nejrealističtější užití, je ve formě informačních tabulí, výloh, voliér a akvárií. To jsou nejčastější prostory, kde se setkáváme s velkými prosklenými panely, jež nás dělí od obsahu za sebou, a tak jsou ideálním prostorem pro zobrazení dodatečné informace a faktů o tomto obsahu. Příklad prosklené voliéry v zoo, třeba v pavilonu goril, by tak mohl návštěvníkům přímo na svém povrchu umožňovat interaktivně zobrazovat detailní edukativní informace o každé z goril a přidaná hodnota interakce s těmito informacemi by mohla sloužit jako přívětivá gamifikace procesu poznávání těchto zvířat.

V této práci bych se tedy po zvážení předešlých možností chtěl dále věnovat spíše kombinaci průhledného displeje a reality se zaměřením na informační rozhraní.

1.3 Rozbor hardwarových nástrojů

O vytvoření 360° projekce se již od počátku práce budou starat 4 projektory s rozlišením až 4K a vysokou svítivostí. Cílem této sekce je nastínit, jak zajistit propojení

a komunikaci zmíněných čtyř projektorů. Zároveň také vyřešení propustnosti skleněné projekční plochy.

1.3.1 Propojení projektorů

Každý z projektorů by měl osvítit čtvrtinu válce a dohromady s ostatními vytvořit jednolitý displej. Potřeba takto velkého, a navíc jednolitého displeje vynucuje propojení a plynulou synchronizaci obrazu u všech čtyř projektorů, ovládané z jedné aplikace. Toho lze docílit dvěma způsoby:

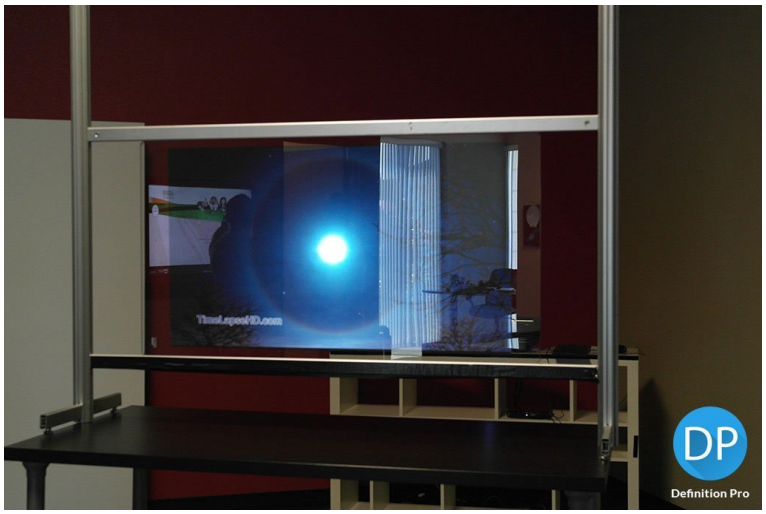
- Přímým kabelovým spojením všech čtyř projektorů do jednoho centrálního počítače s dostatečně výkonnou grafickou kartou (či kartami), podporující zapojení tolika monitorů. Značným problémem takovéhoto řešení je fyzické omezení počtu připojitelných monitorů/projektorů v případě, že bychom chtěli projekční systém dále rozšiřovat.
- Vytvořením komunikující sítě menších jednotek, které budou každá ovládat jeden projektor. To nám umožní vygenerovat škálovatelné prostředí, jež nebude po hardwarové stránce nijak limitováno. Zmiňované jednotky ovládající projektory, musí umět zobrazovat obraz na projektoru a zároveň komunikovat s ostatními jednotkami, případně s centrálním počítačem. Komunikace by v takovém případě byla zajištěna společným zapojením do jedné lokální sítě LAN.

1.3.2 Projekční plocha

Jelikož se jedná o čisté sklo, které má samo o sobě pouze minimální schopnost zachytit na svém povrchu přicházející projekci z projektoru, bude nutné přidat mezi sklo a projektor další průhlednou vrstvu s lepšími projekčními vlastnostmi.

S problematikou projekce na sklo mohou pomoci například komerčně prodávané projekční folie. Ty, při zachování stále dobré průhlednosti, umožňují zachytit projekci přímo na povrchu skla. Jednou z takových folií je fólie od firmy ProFilms, která oplývá dle specifikace výrobce dobrými projekčními vlastnostmi i během denního osvětlení. [3] Tato folie je ale celkem nákladná a ve spojitosti s rozměry projekční plochy válce pro tuto práci nevhodná.

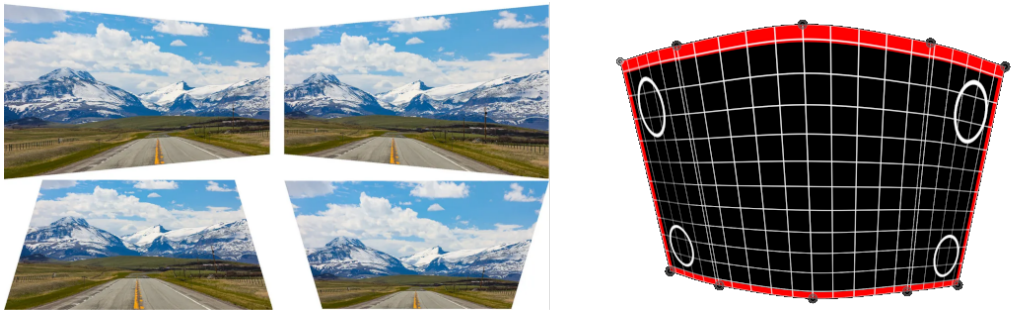
Jako náhradu při prototypovém řešení aplikace lze použít výrazně levnější přístup v podobě obyčejné bílé síťoviny/tylu, skrze kterou je stále dobře vidět a projekce se na ní i tak zachytí. Cena takového prototypového řešení pak nemusí přesáhnout tisíc korun.



Obrázek 1.3. Příklad projekční folie od firmy ProFilms (převzato z [3]).

1.4 Rozbor nástrojů pro projekci

Jelikož se u tohoto displeje jedná o nestandartní projekci na cylindrický (válcovitý) podklad, je třeba vytvořit korekci deformace projekce tak, aby projekce na povrchu válce nevytvářela deformované obrazy. Tyto deformované obrazy mohou mít podobu zakřivených a protáhlých ploch a linií. U projekce na rovnou projekční plochu (dále už jen jako plochá projekce) s tímto nežádoucím efektem pomáhá přímo od výrobce dostupné nastavení projektoru, kde lze pomocí funkce keystone upravit projekci tak, aby i při pozicování projektoru z ostrého úhlu vůči projekční ploše (například ze stropu těsně u plátna), byla výsledná viditelná projekce bez deformace. U ploché projekce mluvíme tedy o lineární perspektivní deformaci obrazu. Pro případ cylindrické projekční plochy, je ale tato metoda nedostačující, jelikož je zde třeba aplikovat krom perspektivní projekce i mapování na válcovou plochu. K tomu je třeba najít softwarové nástroje, které obě korekce budou umožňovat.



Obrázek 1.4. Vlevo různé druhy perspektivní lineární deformace pomocí funkce keystone. Vpravo nelineární perspektivní deformace pro projekci na válec.

Další důležitou funkcí, jež je třeba, je možnost pracovat s více obrazovkami a synchronizovat obraz mezi nimi tak, aby vznikla jednolitá projekce. Navíc důsledkem toho, že se jednotlivé projektory na válci budou pravděpodobně překrývat, by byla vhodná i funkce prolnutí hran obrazu (anglicky edge blending), která umožní pomocí lineárního přechodu ztmavit kraje obrazu v místech, kde se budou projektory překrývat a dosáhnout tak rovnoměrného jasu po celé ploše projekce.

Souvisejícím faktorem, který bude pro použitý software důležitý, je otevřenost tohoto softwarového nástroje, možnost programovatelnosti vlastních uživatelských rozhraní a vytváření nestandartních interakcí s uživatelem. Pro pozdější využití systému dalšími uživateli a výzkumníky je také důležité, aby bylo možné dát výsledný software k dispozici jako open source, aby byl znovupoužitelný a upravitelný pro jiné účely.

Samostatnou kapitolu pak tvoří přístupy k interakci uživatele s projekcí. Klasické ovládací metody jako jsou myš a klávesnice nejsou vhodné pro stojícího a volně se pohybujícího uživatele, protože by limitovali jeho možnosti interakce. Vzhledem k předpokládaným možnostem reálného užití a výhod oproti jiným systémům, je zde vítaná snaha zbavení se jakéhokoliv příslušenství, které by si uživatel musel nosit s sebou.

V následujících podsekcích jsou rozebrány vhodné softwarové nástroje pro řešení problémů zmíněných v odstavcích výše.

1.4.1 Processing

Processing je flexibilní multiplatformní softwarový framework určený pro snadné vytváření rychlých programovatelných bloků, tzv skic (anglicky sketch), a pro první snadné seznamování s programováním. Skládá se z vlastního IDE (uživatelského rozhraní pro programování), kompilátoru a zobrazovacího okna, ve kterém dokáže zobrazit 2D i 3D obsah. Originální verze tohoto programu je psána v jazyce Java a ve stejném jazyce se i programují výše zmiňované skici. I když je Processing v základu psaný v jazyce Java, nabízí zároveň také svou vlastní verzi pro jazyk Python a Javascript. Dále nechybí ani podpora programování pro mobilní platformu Android.

Základním stavebním kamenem, a zároveň silnou stránkou Processingu, jsou jeho knihovny, které snadno a rychle rozšiřují dostupné funkce a umožňují programátorovi rychle získat přístup například k síťové komunikaci pomocí protokolů TCP či OSC, nebo ke knihovnám určeným pro práci se zvukem či dokonce k práci s počítačovým viděním.

Existují také komunitní knihovny, jež vytvořili a zpřístupnili samotní uživatelé Processingu a právě mezi těmito knihovnami lze nalézt takové, které se hodí pro náš případ. Jedná se o knihovny:

- **Keystone** [4] je knihovna umožňující základní lineární perspektivní deformaci pomocí ručního nastavování (posouvání) jednotlivých bodů. Nemá implementovaný edge blending a podpora více monitorů zde sama o sobě také není přímo zabudována. Tu lze ale v Processingu samostatně naimplementovat.
- **SketchMapper** [5] je knihovna umožňující vytváření lineárních i nelineárních perspektivních deformací obrazu. Pro nelineární křivkové deformace umožňuje v jednotlivých bodech nastavovat i směr a sílu působení, a tak měnit tvar hraniční bezzierovi křivky. Stejně jako předchozí případ, neoplývá přímo vlastní integrací podpory více monitorů a také sama od sebe nepodporuje edge blending.

Další možnou knihovnou pak může být knihovna **p5.mapper** [6], která je určena pro Javascript verzi Processingu [7]. Ta ale umožňuje pouze práci s vícebodovou lineární deformací obrazu a jelikož běží ve webovém prohlížeči, nenabízí podporu více monitorů a ani funkci edge blending.

Důležitým aspektem Processingu je také míra svobody, kterou nabízí, a tedy možnost vytvořit kompletní uživatelské rozhraní i s podporou různých druhů interakce. Více o tomto frameworku lze nalézt na webových stránkách frameworku [8].

1.4.2 OpenFrameworks

Jedná se, podobně jako u předchozího Processingu, o multiplatformní softwarový framework, který se snaží zjednodušit a urychlit práci neprofesionálním programátorům a umělcům při vytváření jejich projektů. Tento framework je založený na jazyku C++ a již v základním nastavení obsahuje všechny důležité prvky potřebné pro nastavení a kompilaci do spustitelného programu. Stejně jako u předchozího zmíněného, spočívá jeho síla v rozšiřujících knihovnách, které umožňují rozšířit schopnosti vašeho programu bez složité implementace kolem.

Knihovny, na které zde cílíme, jsou následující:

- **ofxWarp** [9] je knihovna umožňující snadné vytvoření plochých lineárních i nelineárních zahnutých druhů deformovaných obrazů. Umožňuje jednoduše přidávat a odebí-

rat kontrolní body, přetáčet a zrcadlit renderovaný obraz a libovolně upravovat pozici kontrolních bodů. Oplývá také hledanou funkcí edge blending.

- **ofxPiMapper** [10] je v mnohém dost podobný knihovně předchozí. Také umožňuje vytvářet různé obrazy s lineární i nelineární perspektivní deformací s důrazem na optimalizaci pro použití na Raspberry Pi. Nenabízí bohužel ale funkci edge blending ani zrcadlení obrazu a je uzpůsoben pro starší verze Raspberry Pi než máme v našem případě k dispozici.

U obou těchto knihoven se předpokládá, že pro sestavu s více obrazovkami bude nutná vlastní implementace využívající základní nástroje OpenFrameworks a tyto knihovny. Více o tomto frameworku lze nalézt na webových stránkách [11].

1.4.3 TouchDesigner a Resolume Arena

Jedná se o již kompletní aplikace určené pro videomaping, z nichž je TouchDesigner zdarma pro osobní použití. Tyto nástroje umožňují komplexní skládání, přípravu a zobrazování profesionálních videomapingových instalací. Oba nástroje disponují funkcí keystone projection pro upravení lineární či nelineární deformace obrazu. Dále také obsahují zabudovanou podporu pro více monitorů a edge blending. Ani jedna z aplikací ale není určena pro systémy na bázi Linuxu a pro účely školní diplomové práce není ani licence pro osobní použití úplně vhodná.

1.5 Rozbor interakčních nástrojů na bázi gest

Nástrojů pro rozpoznávání lidské ruky a gest existuje celá řada. Ve většině případů se ale skládají ze dvou kroků, jimiž soue samotné rozpoznání objektu ruky spolu s vyhodnocením pozic jednotlivých prstů a v následném kroku vyhodnocení gesta z ze získaných dat.

Polohu ruky a prstů lze obecně sledovat a zkoumat buď pomocí speciální rukavice se senzory snímajícími úhly jednotlivých prstů a natočení ruky [12], jež lze vidět na obrázku 1.5, nebo pomocí metod počítačového vidění. Jelikož je primární snahou této práce snažit se vyhnout nutnosti přidávat uživateli jakýkoliv další nástroj či předmět do rukou, kterým by musel rozhraní ovládat, budeme se dále soustředit na využití metod počítačového vidění.

Obrázek 1.5. Příklad rukavice pro snímání gest ruky. [13]

Metody počítačového vidění pracují na bázi získávání obrazu z různých druhů kamer a následného zpracování získaných obrazových dat do interpretovatelné podoby. Kamery mohou být klasické RGB, jako jsou běžně dostupné webkamery laptopů, hloubkové kamery, jež používá například známý herní systém Xbox Kinect, dále pak infračervené kamery pracující v pro lidské oko neviditelném spektru a stereo kamery využívající skládaní obrazu z více kamer.

Nejpoužívanějšími metodami počítačového vidění pro rozpoznávání lidských rukou jsou:

- **Rozpoznávání barvy** - jedním možným přístupem k této metodě je využití speciální barevné rukavice, která usnadní senzoru rozpoznání pozice ruky, dlaně a jednotlivých prstů. Dalším, a ne málo využívaným přístupem u této metody je detekce barvy lidské kůže z obrazu, oddělení této barvy a vyhodnocení získaného tvaru. Tato metoda může být do značné míry ovlivněna okolním osvětlením, volbou pozadí které neobsahuje stejnou barvu jako je barva kůže, nebo i šumem v obrazu.
- **Rozpoznávání vzhledu** - tato metoda pracuje na principu rozpoznávání tvaru a textury ruky, a následného porovnání s vytvořeným předtrénovaným modelem, na jehož základě vyhodnocuje vzájemnou podobnost. Výhodou této metody je nezávislost na okolním osvětlení.
- **Rozpoznávání na základě pohybu ruky** - lze ji ve většině případů rozdělit do dvou fází. První fází je rozpoznaní ruky v obrazu metodami pro odlišení ruky od pozadí a získání jejího tvaru. V druhé fázi se zapojují algoritmy pro sledování pohybu ruky, ze kterého se vypočte pozice a orientace ruky. Tato metoda je vhodná pro sledování gest v reálném čase ale má nevýhody v podobě citlivosti na změnu osvětlení a na předměty a pozadí stejné barvy jako má lidská kůže.
- **Rozpoznávání založené na kostře ruky** - v této metodě se algoritmus snaží napasovat do obrázku ruky model kostry lidské ruky. Na základě různých parametrů, jako jsou úhly mezi prsty, zakřivení v kloubech, prostor mezi klouby a s pomocí nastavených podmínek, jež musí kostra splňovat, dokáže algoritmus vyhodnotit pozici simulované kostry v obrazu a je díky tomu schopen sledovat nejen pozici jednotlivých kloubů kostry, ale taky převést vnímání této kostry do 3D. Na získané kostře lze poté provádět snadnější vyhodnocování jednotlivých gest. Novější modely pro vyhodnocování 3D pozice kostry ruky pak využívají výhod hlubokého učení (anglicky deep learning) a složitějších neuronových sítí. V tomto případě lze využít jak hloubkové kamery jako je již dříve zmíněný Kinect, ale také klasikou RGB kameru. Sítě k tomu účelu využívané jsou konvoluční neuronové sítě, které mají několik konvolučních vrstev nučených k nalezení různých rysů ruky a propojených vrstev naučených ke klasifikaci gest na základě dříve získaných rysů/parametrů. Tato metoda může být trénována na různých datasetech snímků ruky a může rozpoznávat různé pohyby a gesta ruky.
- **Rozpoznávání založené na hloubce** - využívá kombinaci hloubkové kamery a klasické RGB kamery. Nejprve se provede segmentace ruky z klasického RGB obrazu, která odliší ruku od pozadí a poté se vyhodnotí 3D pozice jednotlivých pixelů ruky z nichž se vypočte geometrický tvar ruky. Na následný tvar se používají další různé metody pro rozpoznávání ruky a gest využívající neuronové sítě či statistické metody. Oproti jiným metodám je rozpoznávání založené na hloubce robustní vůči změnám osvětlení a lze ho použít i v tmavých prostředích.
- **Rozpoznávání založené na 3D modelu** - zakládá se na snaze porovnávání 2D projekce virtuálně vytvořeného 3D modelu ruky se získaným obrazem reálné ruky. 3D model je různě pozicován, natáčen a porovnáván s obrazem. Tato metoda potřebuje často velký

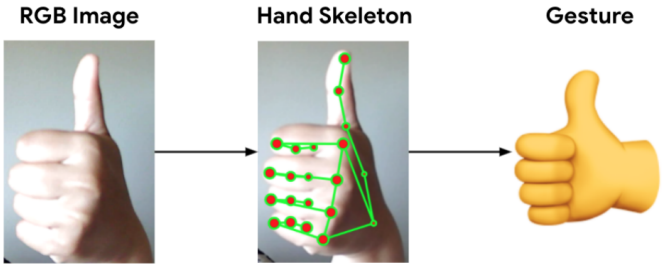
dataset obrázků ruky pro vytvoření počátečního modelu ruky a následný porovnávací proces může být také časově a výpočetně náročný.

Více o jednotlivých metodách, srovnání jednotlivých přístupů a jejich výsledků lze nalézt v článku [14]

1.5.1 Tensorflow hand pose a hand gesture recognizer

V současnosti asi jedna z nejpokročilejších a nejrozšířenějších implementací pro rozpoznávání ruky a gest v reálném čase na běžných zařízeních (mobilní telefon, PC a laptop) je model pro rozpoznávání ruky představený a dostupný v knihovně Tensorflow. Tento model používá k rozpoznávání ruky běžnou RGB kameru, jež je součástí každého laptopu, mobilního telefonu či webkamery u stolních počítačů.

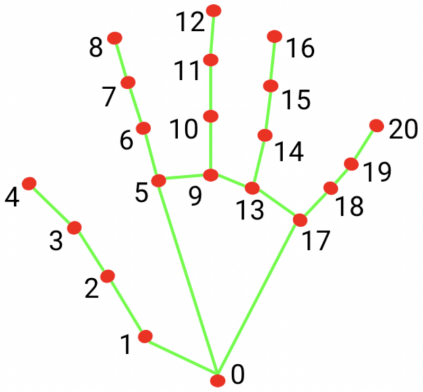
Jak již může napovědět obrázek 1.6 model se skládá z modulu sledování pozice kostry a z klasifikátoru gest. Model sledování je založen na metodě rozpoznávání kostry ruky, zmiňované již výše. Navíc je tato metoda upravena změnou vektoru, kterým se vyhodnocuje natočení a orientace ruky, z původního vektoru mířícího z prvního kloubu prostředníčku do zápěstí, na vektor vycházející ze stejného kloubu, ale mířícího do nového imaginárního bodu. Ten je dán součtem původně používaného vektoru a vektoru tvořeného klouby ukazováčku a malíčku. Dále je metoda pro vyhodnocení kostry ve 3D vylepšena použitím statistického a realistického GHUM modelu lidské ruky pro přesnější vyhodnocení 3D pozic všech kloubů. Výstupem z této části je tedy seznam jednotlivých kloubů (jejichž pojmenování o rozdělení lze vidět na obrázku 1.7 a jejich pozice ve 3D prostoru vůči zápěstí, a vůči kameře v prostoru kamery. Druhou částí je pak vyhodnocení gest ze získané kostry ruky, jež je zjednodušeně založeno na klasifikaci úhlů mezi jednotlivými klouby kostry. Více o zmiňovaném modelu lze nalézt v tomto článku [15]



Obrázek 1.6. Znázornění průběhu rozpoznání gesta v modelu z knihovny TensorFlow. (převzato z [15])

Tento model je implementován a používán v různých knihovnách a je přístupný v několika formách. Za zmínku stojí především tyto:

- **TensorFlowJS**[17] je verzí TensorFlow uzpůsobenou pro fungování ve webových prohlížečích a mobilních zařízeních. Ve webových prohlížečích lze tento model využívat v jazyce Javascript, a to ať už v jeho čisté formě po nalinkování scriptů této knihovny, či například ve webové aplikaci využívající framework React.
- **Mediapipe gesture recognizer**[18] je knihovna dostupná pro Android, Python i Javascript. Dříve zmíněný model z TensorFlow je zde také dostupný. V základu je naučen na sedm jednoduchých gest. Je jej ale možno přeučit i na další gesta.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Obrázek 1.7. Seznam a přiřazení jednotlivých identifikátorů kloubů ruky. (převzato z [16])

- **Ml5 handpose**[19] je součástí knihovny Ml5, která je určitou nadstavbou nad knihovnou TensorFlowJS sloužící k jednoduššímu užití jejich modelů ve webových aplikacích a především pak v symbióze s webovou verzí frameworku Processing s názvem p5.js. Tato varianta umožňuje ale pouze získání kostry ruky, gesta již dále nerozpoznává.

1.6 Rozbor interakčních nástrojů na bázi hlasového ovládání

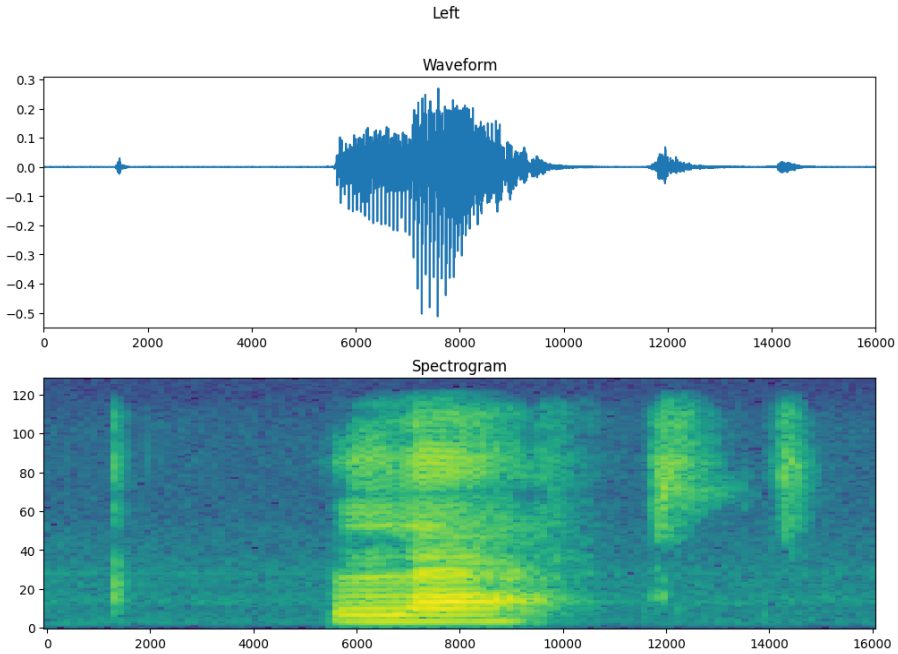
Ovládání hlasovými příkazy je další variantou pro umožnění interakce uživatele s rozhraním na takto velké obrazovce, bez nutnosti držet v ruce a ovládat jakékoliv další nástroje. Existuje mnoho nástrojů a již kompletních řešení, které dokáží rozpoznávat a interpretovat lidský hlas. Mezi nejznámější se pak bude řadit například Amazon Alexa, Google Assistant či Apple Siri. Tito asistenti jsou přizpůsobeni k přijímání a vykonávání jednoduchých i komplexnějších hlasových příkazů a do určité míry i k vedení konverzace. Základním postupem při zpracování hlasového příkazu u těchto asistentů je zaznamenání hlasového vstupu, převedení řeči na text a analýza získaného textu. Na základě analyzovaného textu pak vyhodnotí nejpravděpodobnější odpověď. [20]

Jelikož se v případě této práce jedná pouze o jednoduché rozhraní bez nutnosti komplikovaných interakcí a příkazů, postačují nám pouze jednoslovné příkazy, které si navíc můžeme poté interpretovat a rovnou zpracovat. Nepotřebujeme tedy rozsáhlé funkce výše zmíněných asistentů, ale pouze rozpoznání řeči a převod na text. Pro tyto účely lze využít jeden z dostupných nástrojů pro rozpoznávání a převod řeči na text. V této práci jsem se tak zaměřil především na následující nástroje.

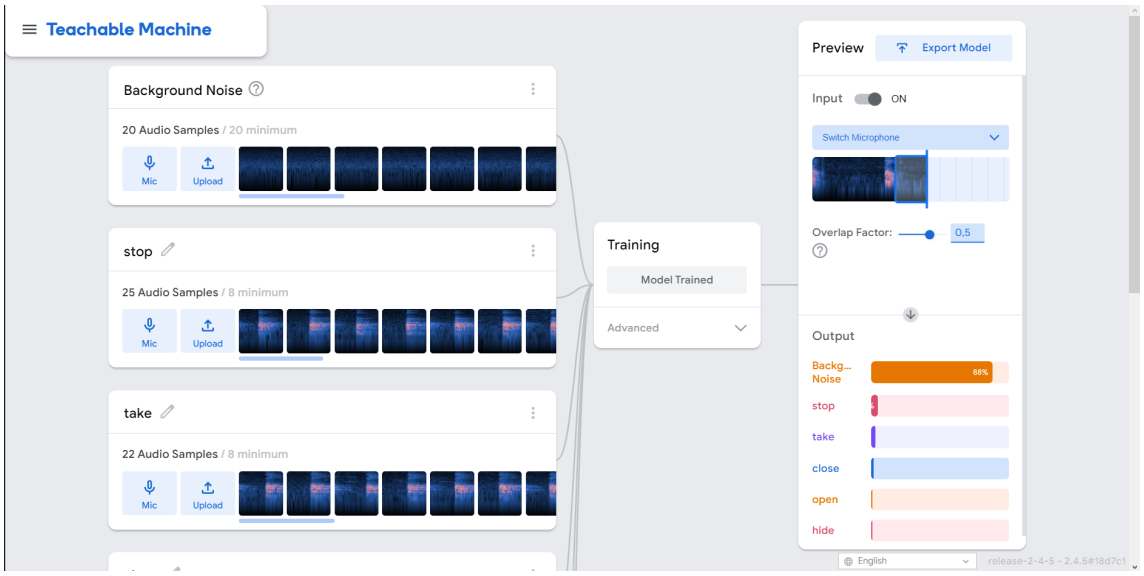
1.6.1 TensorFlow - speech commands

Je další z řady předtrénovaných modelů, jež nabízí knihovna TensorFlow. Tento model funguje na bázi konvoluční neurální sítě natrénované na předem sestaveném seznamu a vzorcích slov. Proto, aby mohla být tato sít použita, je třeba nahrané vzorky řeči převést z klasické časové domény do domény frekvenčního spektra a vytvořit tak jejich spektrogram, jak lze vidět na obrázku 1.8. Tento spektrogram se z původních dat získá použitím Krátkodobé Furierovy transformace [21]. S výsledným spektrogramem už lze jednat jako s obrázkem a vložit jej jako vstup pro konvoluční neuronovou sít, která nám na výstupu předá pravděpodobnosti jednotlivých slov ze seznamu. Více o principu tohoto modelu lze najít v návodném článku, jak takový model sestavit [22].

Tento model lze také snadno učit konkrétním slovům a specifickým podmínkám šumu v pozadí. Tréning takového modelu lze uživatelsky přívětivě provést pomocí webového nástroje Teachable Machine [23], v němž se jednoduše vytvoří jednotlivé třídy reprezentující každé slovo, do nich se nahrají vzorky řeči odpovídající těmto slovům a následně se provede natrénování modelu. Ten může být poté uložen v cloudu a zpřístupněn přes odkaz, a nebo jej lze exportovat do stáhnutelné podoby. Ukázku z rozhraní webové aplikace Teachable Machine lze vidět na obrázku 1.9.



Obrázek 1.8. Ukázka spektogramu získaného ze záznamu v časové doméně. (obr převzat z [22])



Obrázek 1.9. Screenshot z webové aplikace Teachable Machine.

1.6.2 Python SpeechRecognition

Jedná se o další z dostupných knihoven pro převod řeči na text, určenou pro jazyk Python. Na rozdíl od předchozího, nenabízí tato knihovna přímo model rozpoznávající řeč, ale dle zvolených parametrů volá různé veřejné API běžně dostupných rozpoznávačů řeči, jako jsou například Google speech recognition, Google Cloud Speech API, Microsoft Azure Speech či OpenAI Whisper.

Knihovna samotná nabízí pouze nástroje pro zaznamenání zvuku, filtraci hluku na pozadí a zapouzdření řečového rozpoznávání do jednoduchého volání. Aplikace využívající tuto knihovnu pak jen čeká na přijetí řečového vstupu, který nahraje a pošle na zvolené API. Z tohoto API získá odpověď ve formě seznamu slov/frází s přiřazenou pravděpodobností. S takto získanými slovy už lze dále pracovat a vyhodnocovat na jejich základě konkrétní příkazy. [24]

1.7 Síťová komunikace

V případě, že pro projekci z více projektorů zvolíme modulární řešení se samostatnými jednotkami, je dobré se také zamyslet nad způsoby jejich komunikace. Zároveň můžeme předpokládat, že pro interakční část práce bude pravděpodobně třeba oddělená aplikace, která bude také muset nějak komunikovat s projekcí.

Co se týče projekce, obecně lze přemýšlet nad dvěma způsoby komunikace. Prvním z nich bude streamování obrazu/videa z řídící jednotky do klientských jednotek u každého projektoru. Na straně řídící aplikace by se skládala celá obrazovka, která by se poté rozdělila na příslušný počet projektorů a výsledný obraz by se zaslal do klientské jednotky. Skládání obrazu by mohlo probíhat pomocí framebufferu [fbo], do kterého by se vykreslila celá obrazovka a poté by se jeho specifické části odeslali klientským aplikacím ve formě obrázku/textury.

Pro samotné posílání obrázku by bylo možné použít například funkce knihovny **ofxStreamer** [26], jež je dalším doplňkem pro framework OpenFrameworks sloužícím k posílání obrázků mezi počítači s využitím kódování H.264.

Druhým způsobem komunikace bude zasílání specifických instrukcí, kterým bude klientská aplikace rozumět a dokáže na jejich základě upravit zobrazovaný obsah. Obdobně bude táke třeba užít posílání instrukcí i v případě komunikace interakční aplikace s projekcí.

Pro posílání instrukcí lze využít exitujících protokolů jako je protokol UDP[28] či v uměleckém průmyslu rozšířený protokol OSC (Open Sound Control)[27].

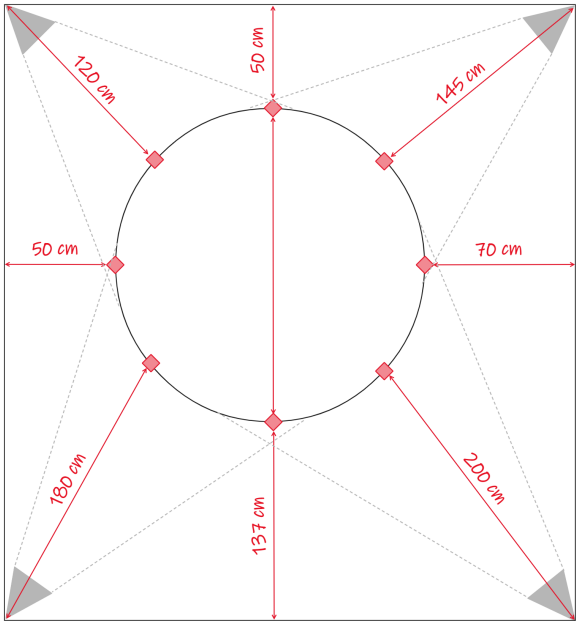
Kapitola 2
Implementace

Na implementaci řešení se podíváme ze dvou stránek. Ze stránky hardwarové, kde bude popsáno řešení rozložení projektorů, promítací plochy a předávání obrazu do projektoru, a ze stránky softwarové, kde budou rozebrány testovaná softwarová řešení projekce i interakcí. Dále bude dopodrobna rozebrána i vybraná varianta s podrobným popisem vnitřní logiky.

Co se týče hardwarového příslušenství a techniky dostupné pro práci, již od zadání práce byly k dispozici 4x 4K projektor, 4x minipočítač RaspberryPi 400, centrální desktopový počítač s odpovídajícím výkonem (NVidia GeForce Titan X, 16GB RAM, Intel ...), televizor, mikrofon a webkamera.

2.1 Prostředí

Na úvod stojí za zmínku popis pracovního prostředí, ve kterém se bude práce dále orientovat. To se nachází v místnosti s obdélníkovým půdorysem cca 4,6x5,6 m. V relativním středu místnosti se nalézá skleněný válec skládající se z 6 posouvatelných skleněných plátů. Válec má průměr cca 2,5 m a výšku 3,2 m. Strop tvoří betonové nosníky, na které je přimontováno zavěšení skla. Jednotlivé vzdálenosti skla od stěn jsou znázorněny na obrázku 2.1. V místnosti se také dále nachází pracovní stůl pro umístění řídícího počítače.



Obrázek 2.1. Schéma vzdáleností skla od hranic místnosti a projektorů.

2.2 Příprava fyzického hardwaru pro projekt

Základní projekce by se měla skládat ze čtyř projektorů umístěných v horních rozích místnosti tak, aby každý projektor zabíral alespoň čtvrtinu válce. Jelikož projektory nelze, díky výše zmíněným stropním traverzám, zavěsit na strop, bylo nutné je umístit na stěnu a umožnit jim rotaci ve vertikální ose, tak aby i při přichycení na stěnu místnosti mohli z rohu směřovat do středu místnosti. Toho bylo docíleno přidáním dalšího stupně volnosti mezi projektor a uchycovací nástavec. Původně pro tuto funkci byl vytvořen 3D model, jež měl být vytištěn na 3D tiskárně a lze ho vidět na obrázku 2.2. Pro ušetření času a vytížení tiskárny bylo ale později zvoleno jednodušší řešení v podobě překližkové desky, do níž jsou vyvrtány otvory pro uchycení projektoru a drážky pro pohyblivé uchycení na nástavec. Samotné uchycení projektoru na nástavec lze vidět na obrázku 2.2.

Nakonec byli plně instalovány a zprovozněny pouze dva projektory pokrývající půlku válce. K tomuto kroku bylo přikročeno z důvodu nevyužití zbylé plochy pro další práci a fakticky chybějícím nástrojům pro uchycení nástavce projektoru do pevné nosné zdi (na rozdíl od sádrokartonu v předchozím případě). I přes omezení jen na dva projektory, je možné rozhraní plně otestovat. Díky modularitě hardwaru a zvolené architektuře aplikace, která umožňuje snadné přidání neomezeného množství nových projektorů, nepředstavuje pozdější navýšení o zbytek projektorů žádný problém.

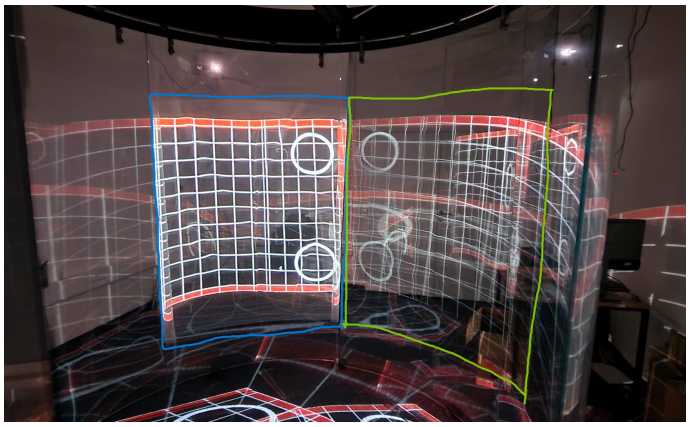
Obrázek 2.2. Vlevo model uchycení projektoru na nástavec s přidáním vertikáního stupně volnosti. Vpravo výsledné uchycení na stěnu.

Proto, aby byla do budoucna zachována možnost rozšíření projekce o další projektory, bylo zvoleno řešení se samostatnými jednotkami u každého projektoru. Jako kontrolní jednotka projektoru byl zvolen dostupný minipočítač Raspberry Pi 400 s operačním systémem na bázi Linuxového jádra. Raspberry se k projektoru připojilo pomocí HDMI kabelu pro posílání obrazu a pomocí sériového kabelu RS-232 pro vzdálené ovládání projektoru, které probíhá skrze speciální příkazy posílané přes zmíněný seriový port. Jednotlivé Raspberry jednotky jsou pak propojeny pomocí lokální sítě LAN s centrálním počítačem. Celou sestavu jednoho projektoru a jeho řídící jednotky lze vidět na obrázku 2.3.

Pro projekční plochu byla jako levná a rychle dostupná varianta zvolena možnost překrytí skla bílou síťovinou. V rámci testování byla vyzkoušena síťovina černá, ta byla sice

Obrázek 2.3. Kompletní projekční jednotka – projektor + minipočítač Raspberry Pi 400.

za denního světla mnohem lépe průhledná, ale projekce se na jejím povrchu zachycovala jen o něco lépe než na čistém skle a zároveň v kombinaci se sklem za ní vytvářela nepříjemné artefakty. Bílá síťovina oproti tomu dovolovala jasné a ostré zobrazení viditelné i z druhé strany, přičemž průhlednost v tmavé místnosti zůstávala stále na dobré úrovni. Pro lepší představu lze porovnat kvality obou projekcí na obrázku 2.4.



Obrázek 2.4. Projekce na různé druhy síťoviny. Modrá oblast - bílá síťovina. Zelená oblast - černá síťovina.

Dovnitř válce byla poté umístěna webkamera a mikrofon. Pro umístění kamery i mikrofonu se v původních úvahách jevilo jako nejlepší zavěšení k horní hraně skla ve středu projekce. Nakonec po otestování, bylo přikročeno k variantě umístit kameru i mikrofon na provizorní vyvýšené místo u spodní hrany projekce. Mikrofon v této poloze přijímal méně rušivých ozvěn způsobených akustikou válce a kamera byla lépe schopna rozpoznávat uživatelovu ruku a gesta. Finální rozložení lze vidět na obrázku 2.5.

Hardwarová sestava je doplněna ještě o další projektor, který simuluje reálnou scénu za válcem projekcí na zeď. V původním plánu bylo využít pouze obrazu televizní obrazovky dostupné již při zadávání práce. Díky svým rozměrům a vzdálenosti od uživatele, se ale televizní obrazovka ukázala jako nedostatečná.

Obrázek 2.5. Umístění kamery a mikrofonu ve válci.

2.3 Softwarová implementace

Jelikož bylo zvoleno modulární řešení pomocí Raspberry Pi, které, jak už bylo zmíněno, využívá systémy na linuxové bázi, odpadla hned na začátku možnost využití programů TouchDesigner či Resolume Arena, jež vyžadují operační systém Windows. Další selektující vlastností se u Raspberry ukázal nepříliš velký grafický výkon, který způsoboval při plném rozlišení značný pokles fps (zobrazených snímků za sekundu). I při snížení rozlišení na FullHD (1920x1080 bodů) měli zbývající softwarová řešení problém s plynulým zobrazením obrazu.

Toto zjištění jsem jednoduše demonstroval a změřil na prosté aplikaci bez jakékoliv vnitřní logiky a složitého vykreslování, která pouze zobrazovala bílý kruh posouvající se každý snímek o jeden pixel doprava. Z výsledků měření, jež lze vidět v tabulce 2.1 je patrné, že se jako lepší a stabilnější varianta ukázal framework OpenFrameworks, který se pohyboval při FullHD rozlišení na stabilních 37 snímcích za sekundu.

Windows 10 (Nvidia GeForce Titan X,..)	Průměr Fps	Min Fps	Max Fps
OpenFrameworks	59.5	55.7	60.2
Processing	59.6	57.3	60.2
Raspberry Pi 400	**Průměr Fps**	**Min Fps**	**Max Fps**
OpenFrameworks	37	34.7	60
Processing	9.5	8.9	33.8

Tabulka 2.1. Výsledky měření výkonu jednotlivých frameworků na testovací aplikaci.

Při hlubším prozkoumání společně s projekčními knihovnami bylo také pozorováno že Processing a jeho SurfaceMapper se pohyboval na stolním počítači (při FullHD) okolo 5 snímků za vteřinu, což způsobovalo nepříjemné trhání obrazu a prakticky znemožňovalo další práci. OpenFrameworks oproti tomu vykazoval stabilních 60 snímků za vteřinu u desktopového PC a 25 fps na Raspberry Pi, což bylo vyhodnoceno jako dostatečné vzhledem k tomu, že i filmy mívají běžně 25fps. Díky těmto dosahovaným parametrům a dostupné knihovně ofxWarp, která obsahuje všechny pro naše řešení potřebné funkce, se stal framework OpenFrameworks vítězem pro další implementaci.

Po hlubším zkoumání bylo zjištěno, že OpenFrameworks nepodporuje (alespoň prozatím) operační systémy s architekturou arm64. Tedy systémy, na kterých primárně běží Raspberry Pi 400. Zároveň není možné použít systémový obraz, který doporučuje a poskytuje rovnou sám OpenFrameworks na svých stránkách, jelikož se jedná o starou verzi systému, která není podporována základní deskou Raspberry Pi 400. Po delším zkoušení se jako nejvhodnější systém pro OpenFrameworks na Raspberry Pi 400 ukázal 32bitový Raspberry Pi OS na verzi Debian 10 (bustler). I když je OpenFrameworks sám o sobě nástrojem nezávislým na platformě, je pro určité aplikace nutné menších úprav v případě, že jako zde u Raspberry, nepoužívají platformy běžné OpenGL ale OpenGL ES. Proto bylo pro správné fungování knihovny ofxWarp, která je vytvořena primárně pro desktopové počítače, potřeba drobně upravit základní nastavení aplikace a přepsat některé funkce užívané v shaderech patřících k této knihovně. Z toho důvodu jsou k této práci přiloženy dvě verze klientské aplikace, jedna pro desktopová zařízení a druhá pro minipočítače typu Raspberry Pi. Díky tomu má tato práce navíc potenciál obohatit využití OpenFrameworks na Raspberry Pi 400.

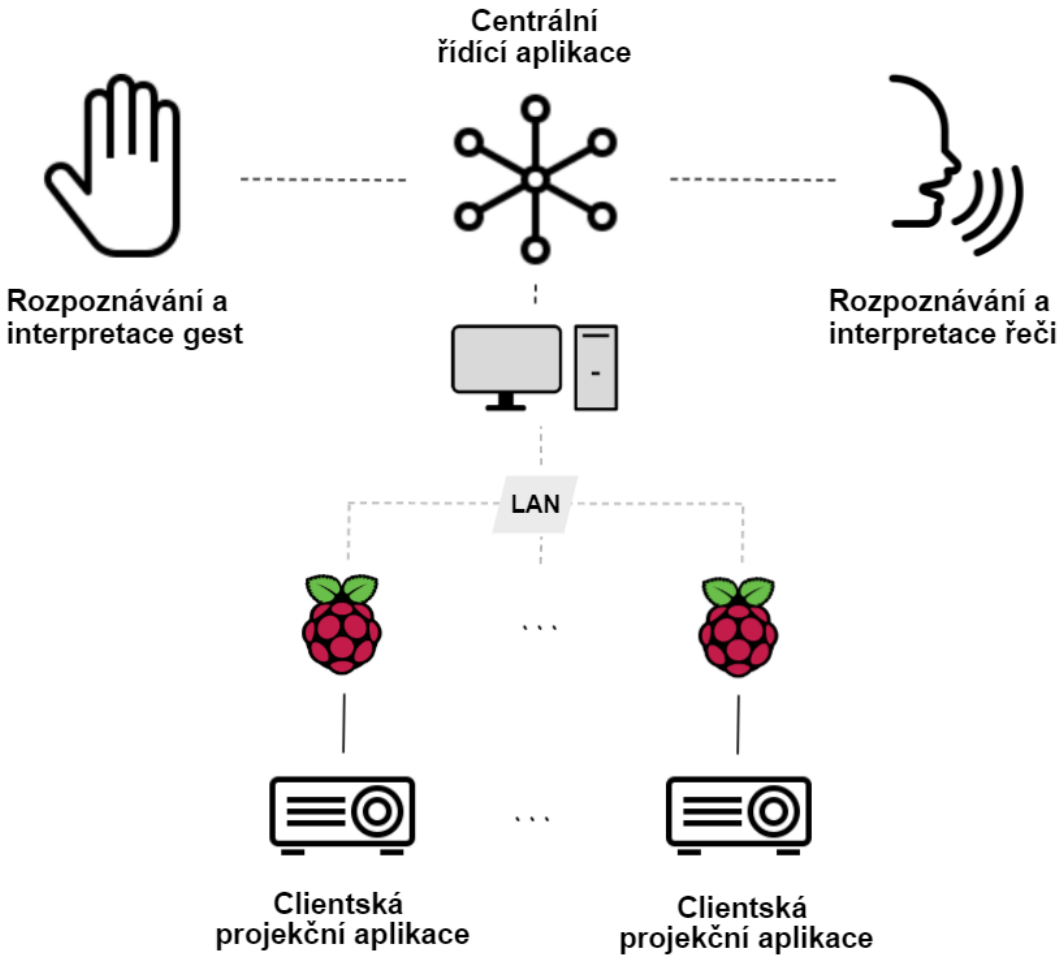
Celé softwarové řešení se skládá z řídící aplikace běžící na hlavním počítači, klientských aplikací běžících na jednotlivých Raspberry Pi a samostatných aplikací pro interakci za pomoci gest a hlasu. Řídící aplikace se stará o definici zobrazovaného obsahu a rozhodování na kterém projektoru se má obsah promítat. Klientské aplikace se starají o individuální deformaci obrazu dle aktuálního projektoru a následné zobrazení obsahu podle pokynů od řídící aplikace. Interakční aplikace se pak starají o zpracování vstupů od uživatele skrze mikrofon a webkameru. Schéma celého řešení lze vidět na obrázku 2.6.

2.3.1 Řízení zobrazování

V obou aplikacích (klientské i řídící) probíhá hlavní smyčka programu, která v každém cyklu projde nejdříve metodu *update()* a pak metodu *draw()*. Metoda update() slouží k aktualizaci logiky aplikace a odesílání či přijímání instrukcí skrze síťovou komunikaci. Metoda draw() pak pouze vykresluje obsah na lokální obrazovku.

Logika samotného zobrazení obsahu probíhá v řídící i klientské aplikaci stejně. V obou aplikacích jsou vytvořeny slovníky sloužící k ukládání informací o jednotlivých grafických elementech. Nový objekt je vždy podle svého typu (obdélník, obrázek nebo text) uložen do slovníku a spárován se svým identifikátorem. Tento objekt může být poté v případě změny parametrů díky svému identifikátoru aktualizován anebo smazán, pokud již nemá být zobrazen. V metodě draw() jsou slovníky iterovány a veškerý jejich obsah je vykreslen na monitor/projektor.

Základními zobrazovanými prvky obrazu jednotlivých obrazovek jsou tedy grafické elementy. Typy těchto elementů, jak už zaznělo výše, jsou obdélník, text a obrázek. Pro každý z nich je na této úrovni definována vlastní struktura ukládající potřebná data



Obrázek 2.6. Schema jednotlivých částí aplikace.

definující pozici, velikost či barvu daného elementu. Dále má každý element vlastní přiřazený identifikátor, dle kterého může být později aktualizován či smazán.

Z grafických elementů jsou tvořeny všechny objekty uživatelského rozhraní. Ty jsou definovány v prostoru imaginární složené obrazovky, a tedy na rozdíl od elementů, které se můžou v případě zásahu do více lokálních obrazovek nacházet ve více obrazovkách, jsou objekty vždy vytvořeny jen jednou a lze s nimi interagovat.

Vzhledem k zvolenému objektově orientovanému přístupu programování je základem všech těchto objektů třída **Object**. Ta obsahuje data a metody společné pro všechny ostatní objekty. Konkrétně se jedná o identifikátor a typ objektu, pozici na imaginární složené obrazovce, šířku a výšku. Z metod pak obsahuje ty sloužící pro aktualizaci pozice elementů, zjištění, zda se v metodě předaný bod nachází nad objektem, či metodu pro aktualizaci elementů tvořících objekt. Každý objekt má také svůj vlastní seznam elementů, ze kterých se skládá. Každý záznam v seznamu je tvořen identifikátorem a typem elementu a na základě těchto informací je skrze třídu Projection controller (jež je více popsána níže) vytvářen a aktualizován na jednotlivých obrazovkách.

Ostatní objekty z třídy Object dědí a přidávají další metody a parametry či přepisují stávající dle upraveného chování. Jejich příklady lze vidět na obrázku 2.7. Patří mezi ně:

- Třída **RectObject** reprezentuje objekt obyčejného obdélníku s nastavitelnou šířkou, výškou a barvou.
- Třída **TextString** reprezentuje objekt jednoduchého textu s nastavitelným fontem a barvou.
- Třída **Image** reprezentuje objekt obrázku. Ten má dva módy. První je čistý obrázek s nastavitelnou velikostí a druhý mód umožňuje u obrázku zobrazit i popisek. Obrázek u popisku lze zavřít a mít viditelný pouze popisek. Druhý z jmenovaných módů se automaticky použije když se při vytváření objektu definuje i text popisku.
- Třída **InfoContainer** Jedná se o komplexnější objekt, sestávající v základu z textové hlavičky a pozadí, které je tvořeno obdélníkem specifikované barvy. K hlavičce lze přidávat další řádky textu, přičemž obdélník v pozadí se automaticky přepočítává, aby obaloval veškerý obsah.

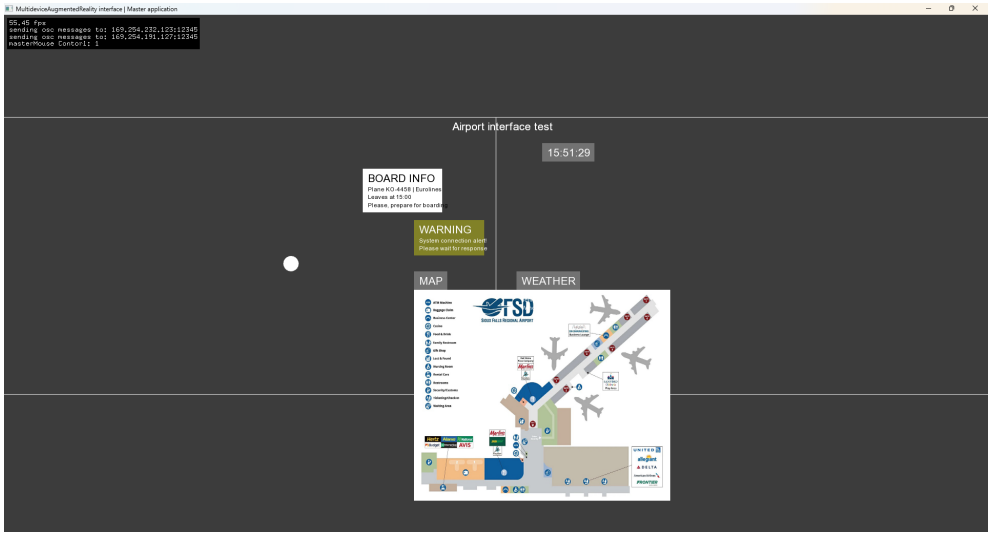
Obrázek 2.7. Ukázky jednotlivých UI objektů - a) Image s popiskem, b) Text, c) InfoContainer, d) RectObject

Všechny objekty uživatelského rozhraní jsou uloženy ve třídě **ObjectsManager**, odkud jsou pomocí identifikátoru přístupné všem dalším třídám a manažerům. Krom toho je zde uložena reference na další třídu **Mouse**, která se stará o zpracování a ukládání informací o aktuální pozici kurzoru na imaginární složené obrazovce. Dále se také stará o převod souřadnic z tohoto prostoru do prostoru lokální obrazovky řídícího počítače a zpět. Díky tomu umožňuje operátorovi nejen ovládat kurzor pomocí klasické myši skrze řídící obrazovku, ale také zároveň uživateli skrze další interakční modality.

Transformaci elementů mezi imaginární složenou obrazovkou a jednotlivými projekčními obrazovkami zajišťuje třída **Projection controller**, která sdružuje veškeré obrazovky do jedné o rozlišení ve vodorovné ose rovnajícím se součtu rozlišení všech obrazovek (v základním nastavení bez překrytí). Dále obsahuje metody pro transformaci, jimž se zadá velikost a pozice na imaginární složené obrazovce a vnitřní logika metod je poté přepočítá do souřadnic konkrétních obrazovek, na kterých se má daný objekt zobrazit. Elementy s takto přepočítanými parametry se posílají každé dotčené obrazovce k uložení.

O ukládání grafických elementů pro každou lokální obrazovku se v řídící aplikaci stará třída **Screen**. Ta dále obsahuje metody pro komunikaci s konkrétní klientskou aplikací

a zároveň slouží k lokálnímu vykreslování obsahu na monitoru hlavního počítače. Pro lokální vykreslování je ve třídách Screen všech obrazovek volána metoda *drawScreen()*. Třída si pro každý element uchovává stav, zda byl od posledního poslání klientské aplikaci změněn/aktualizován či nikoliv. Pokud k aktualizaci došlo, jsou informace o každém takovém elementu odeslány do klientské aplikace při nejbližším volání metody *updateClients()*. Komunikace je optimalizována tak, aby se všechny aktuální příkazy klientům poslaly ve stejném cyklu hlavní aplikační metody update(). Screenshot z reálné obrazovky řídící aplikace lze vidět na obrázku 2.8.



Obrázek 2.8. Screenshot obrazovky z řídící aplikace.

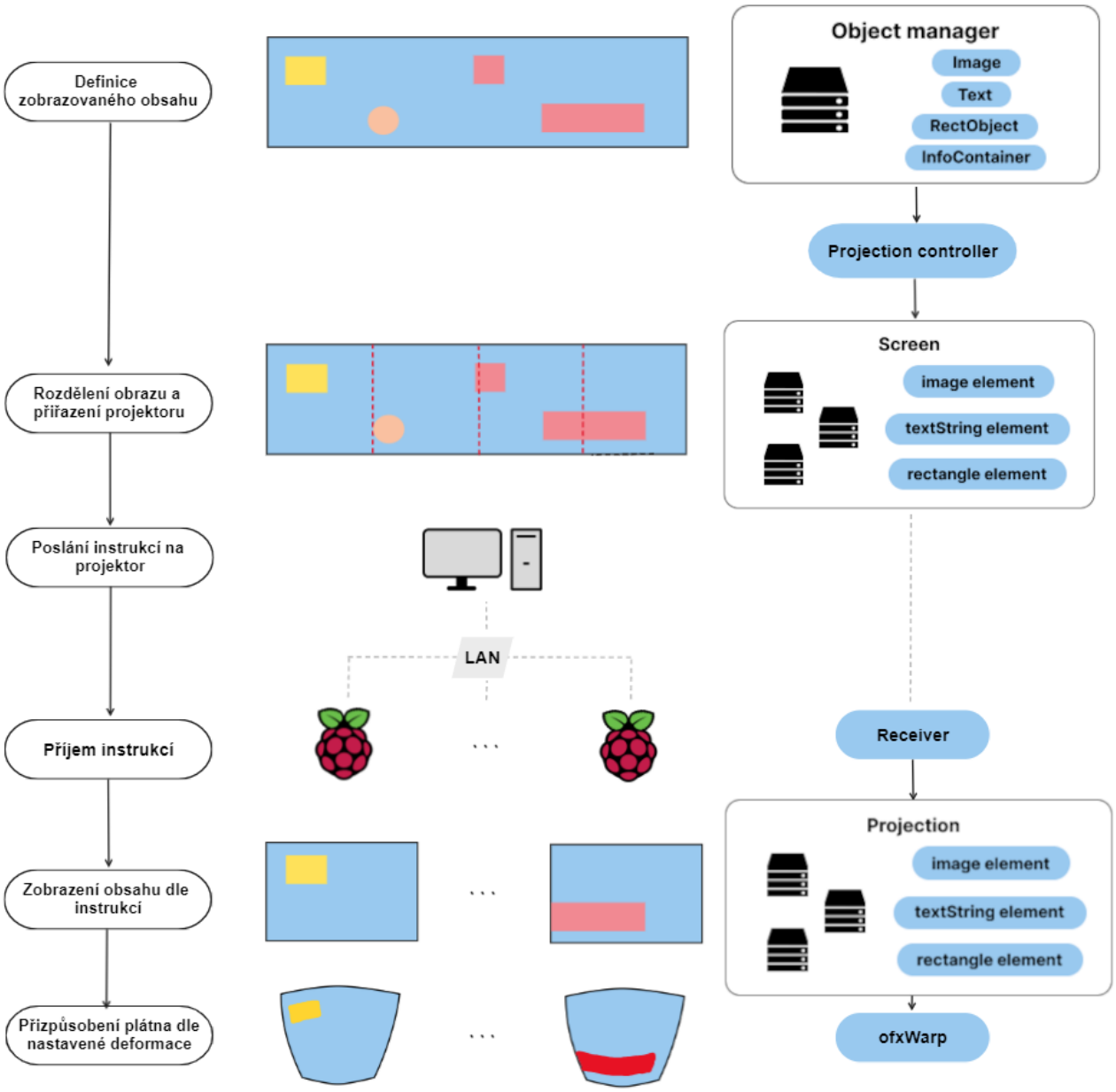
Na straně klientské aplikace se o přijetí instrukcí z řídící aplikace stará třída **Receiver**. Ta jednoduše vytvoří posluchače na specifikovaném portu a přijímá všechny zaslané zprávy. V metodě *processReceivedMessages()*, volané při každém cyklu metody update() klientské aplikace, jsou všechny přijaté zprávy zpracovány a je dle jejich instrukcí změněn stav elementů uložených ve třídě Projection.

Třída **Projection** je obdobou výše zmíněné třídy Screen v řídící aplikaci. Slouží k ukládání všech zobrazovaných elementů a taktéž k jejich vykreslení na obrazovku projektoru voláním metody *drawProjection()*. Zároveň na základě předaných instrukcí zobrazuje či skrývá virtuální kurzor myši.

V obou aplikacích se ve spojitosti s vykreslováním textů používá třída **FontManager**. Ta v sobě ukládá všechny načtené velikosti použitých fontů a obsahuje metody pro převod z typeFont výčtového typu (s jehož pomocí se informace o použitém fontu posílá skrze síť) na skutečný ofTrueType font a naopak.

Obrázky se skrze OSC protokol neposílají, pouze relativní cesta k jejich souboru, který musí být přítomen u řídící i klientské aplikace a umístěn na stejném místě v adresáři. Tato cesta byla zvolena z důvodu optimalizace rychlosti přenosu. Při testování posílání obrázku po síti, se ukázalo, že přenos zpomaluje interakci, a proto jsme přistoupili k nahrání obrazového obsahu dopředu. V budoucnu plánujeme tento problém odstranit vhodnou kompresí obrazu.

Deformace obrazu v klientské aplikaci je samostatně popsána v sekci 2.3.3. Schéma celé zobrazovací pipeline lze vidět na obrázku 2.9.



Obrázek 2.9. Schema jednotlivých částí aplikace.

2.3.2 Síťová komunikace

Síťová komunikace probíhá pomocí protokolu OSC. OSC se běžně používá pro propojování multimediálních zařízení a kontrolerů v hudebních a jiných uměleckých představeních. Jeho výhodou, například oproti klasickému TCP/IP protokolu, je možnost snadného definování konkrétních zpráv a jejich parametrů a velká míra podpory dalšími softwary. Každá odesílaná a přijímaná zpráva je definovaná svou specifickou adresou. Každé takové adrese může programátor přiřadit libovolné parametry. Ty jsou na straně odesílatele do zprávy vloženy a na straně příjemce zase, dle konkrétní adresy, ze zprávy vyjmuty. Příklad takové zprávy posílající pozici myši je vypsán níže.

```
void Screen::sendMousePosition() {
   displayCursor = true;
   ofxOscMessage m;
```

```
    m.setAddress("/mouse/position");
    m.addFloatArg(mouseX);
    m.addFloatArg(mouseY);
    sender.sendMessage(m, false);
}
```

Jak už bylo zmíněno v předchozích kapitolách, systém komunikace se zakládá na synchronizaci slovníků s grafickými elementy mezi řídící aplikací a přiřazenými klientskými obrazovkami. Kromě toho je samostatně odesílána aktuální pozice a stav kurzoru a také příkazy pro vyčištění obrazovky či vymazání jednotlivých elementů. Nové elementy jsou nejprve vloženy do lokálních slovníků řídící aplikace, jsou jim nastaveny příznaky pro odeslání a poté jsou veškeré elementy s aktivními příznaky přeposlány do klientské aplikace která si je uloží. V případě aktualizace stavu elementu je opět nejdříve aktualizován element v lokálním slovníku a nastaven jeho příznak a poté, stejně jako u nového elementu, je zaslán klientské aplikaci k přepsání.

V rámci zasílaných parametrů bylo třeba vytvořit několik pomocných funkcí. Knihovna ofxOsc, která je součástí OpenFrameworks, sice umožňuje zasílání rgba barvy, ale ve formátu nekompatibilním s typem ofColor používaným v aplikaci. Proto je barva nejprve převedena do textového formátu, ve kterém se pošle klientské aplikaci a zde je následně znovu sestavena do původního typu. Podobně bylo třeba zasílání konkrétního fontu nahradit zasíláním pouze jeho typu v numerické hodnotě odpovídající položce z výčtového typu *FontType* dostupnému na obou aplikacích.

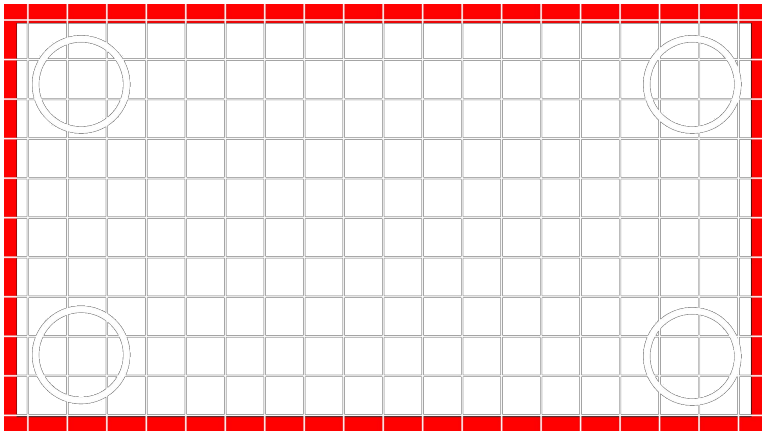
2.3.3 Nastavení a řízení projekce

S celým procesem nastavení deformace projekce pomáhá knihovna/addon ofxWarp. Ta umožňuje vytvořit objekt obrazovky nazývaný warp. Na tento objekt se poté promítá veškerý obsah. OfxWarp umožňuje použití tří základních druhů deformací, které jsou:

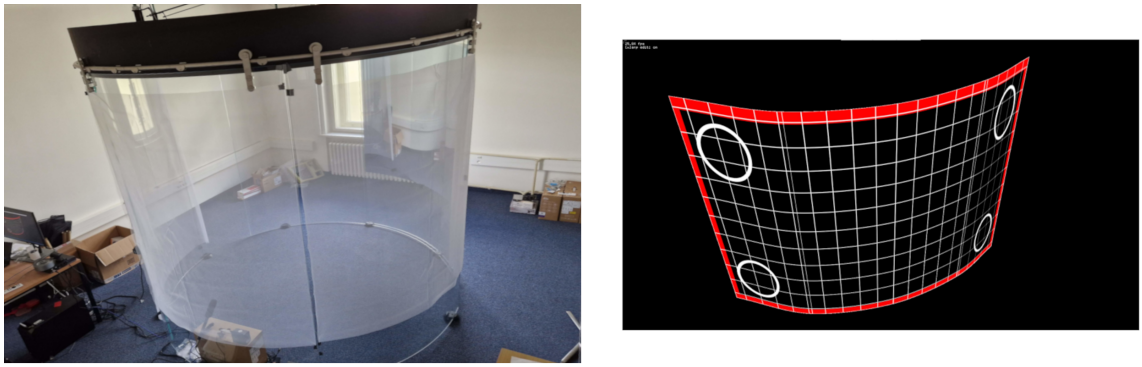
- Perspektivní warp – slouží pro kompenzaci deformace obrazu v případě projekce na rovnou plochu pod ostřejším úhlem.
- Bilineární warp – slouží pro kompenzaci zakřivení způsobeného projekcí na zaoblenou plochu. Předpokládá se, že projektor je umístěn kolmo proti projekční ploše.
- Perspektivně-bilineární warp – kombinuje oba předchozí přístupy. Umožňuje kompenzovat deformaci obrazu způsobenou zakřivením projekční plochy a zároveň ostrým úhlem projekce.

Z výše zmíněných variant deformací byla v aplikaci užita varianta perspektivně-bilineárního warpu. Nastavení deformace se za běhu aplikace provádí zmáčknutím klávesy *w*. V tomto režimu lze jednoduše přesunem kontrolních bodů na nové pozice upravit výstupní deformaci. Pomocí akčních kláves F1-F12 lze přidávat či ubírat kontrolní body a také rotovat či zrcadlit obraz. Pro lepší nastavení deformace na válci je v tomto režimu zobrazena speciální čtvercová kalibrační síť, kterou lze vidět na obrázku 2.10. Veškeré nastavení se po ukončení aplikace ukládá do externího JSON souboru. Výsledný tvar a deformace obrazu jednoho z projektorů lze vidět na obrázku 2.11.

Pro správné zobrazení obsahu na ploše válce je třeba také brát v potaz nutnost zrcadlení celého obsahu. Díky tomu, že projektory jsou umístěny z vnější strany válce a uživatel stojí uvnitř, viděl by uživatel bez jakékoliv korekce rozhraní chybně zrcadlené. Tedy pravou stranu nalevo a levou napravo. Pro kompenzaci tohoto problému stačí invertovat pořadí klientských obrazovek a na všech aplikovat zrcadlení obrazu podle vertikální osy,



Obrázek 2.10. Textura použitá pro kalibrační síť v režimu nastavení deformace.



Obrázek 2.11. Vlevo pohled z pozice projektoru. Vpravo snímek obrazovky s nastavenou deformací na připojeném Raspberry Pi.

které v základu umožňuje knihovna ofxWarp. Tímto způsobem dostaneme celkovou projekci, která je vůči uživateli správně orientována.

2.3.4 Interakce pomocí rukou

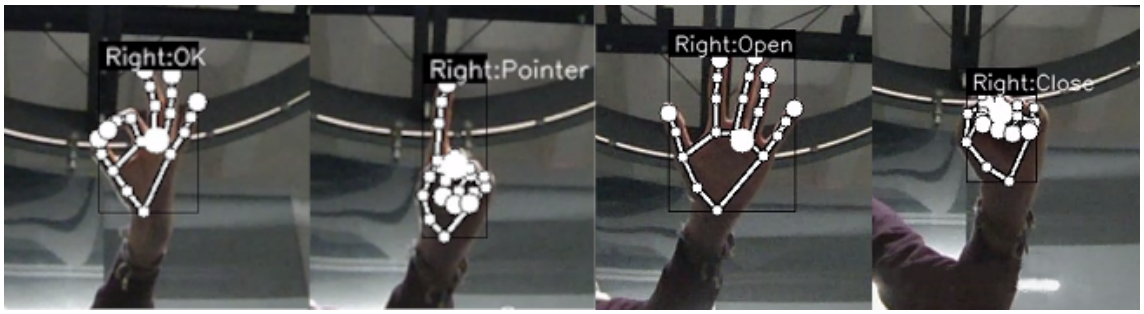
V kapitole věnující se rozboru interakce pomocí gest a rukou bylo rozebráno několik dostupných nástrojů. V rámci implementace bylo vyzkoušeno více ze zmíněných, a proto v této kapitole budou dále rozebrány všechny vyzkoušené varianty. Ve všech případech se jedná o samostatné aplikace, které nejsou nijak přidružené k řídící aplikaci a komunikují s ní jen pomocí již dříve zmíněného OSC protokolu. Výhodou tohoto přístupu je nejen oddělení aplikací tak, že každá může na lokálním počítači běžet na jiném jádře, ale také případná možnost pustit tuto interakční aplikaci úplně na jiném zařízení, které bude s hlavním počítačem pouze síťově spojeno. V souvislosti s kvalitou rozpoznávání ruky a gest jsou zde také rozebrány různé vyzkoušené možnosti kamer a osvětlení.

Hlavní myšlenkou interakce pomocí rukou je umožnit uživateli ovládat rozhraní pouze za použití gest a pozice ruky. Pozicí ruky vůči obrazovce by uživatel ovládal pozici kurzoru a gesty pak uchopoval či pouštěl objekty, či jinak interagoval s prvky uživatelského rozhraní. V rámci této myšlenky byla jako první testována varianta aplikace pro jazyk Python vycházející z volně dostupného projektu hand-gesture-recognition-using-mediapipe [29], která využívá k rozpoznávání rukou model Mediapipe hands a umožňuje dále klasifikovat předem naučená gesta jako je otevřená a zavřená ruka, pointing a gesto OK (otevřená ruka poze se spojeným palcem a ukazováčkem).

Poté co aplikace rozpoznala ruku na obraze kamery a vyhodnotila gesto které ruka dělá, byl dle zobrazeného gesta vyhodnocen konkrétní příkaz a následně zaslán do řídící aplikace. Interpretace gest na příkazy v tomto případě byla následující:

- **Pointing** – zobraz kurzor
- **Zavřená ruka** – uchop objekt
- **Otevřená ruka** – pusť objekt

Nerozpoznané gesto nebo přechod na jiné než uchopení, vedlo k automatickému puštění objektu. To mělo během testování této varianty negativní dopady na uživatelskou zkušenost, jelikož takové výpadky nebyly za běhu aplikace ojedinělé. Z toho důvodu bylo poté přistoupeno pouze k variantě dvou příkazů a to *uchopit* pro zavřenou ruku a *pustit* pro otevřenou ruku. Kurzor byl tak zobrazován neustále.



Obrázek 2.12. Obrázky všech rozpoznatelných gest.

MediPipe hands k 3D pozicím kloubů ruky přiřazuje i 2D pozici v obraze kamery. Díky tomu bylo možné stanovit pozici ruky na kameře vůči hornímu levému okraji obrazu kamery, tu následně transformovat do pozice vůči středu obrazu kamery a odeslat do řídící aplikace.

Model ale nebyl bezchybný a s rostoucí vzdáleností od kamery měl větší tendenci ztrácet sledovanou ruku, díky čemuž docházelo k poskakování kurzoru. Z toho důvodu byla na straně interakční aplikace implementována pomocná třída **Mover**, která slouží k interpolaci pohybu tak, aby byl jeho pohyb plynulý.

Třída si vždy pamatuje aktuální pozici kurzoru a nový cíl, který je daný pozicí ruky. K tomuto cíli postupně směřuje v závislosti na aktuální rychlosti vycházející ze zrychlení daného vzdáleností staré pozice od nového cíle. Aktuální pozice se poté aktualizuje každý cyklus aplikace. Místo skokového posunu kurzoru tak došlo k rozložení posunu do delšího časového úseku a zjemnění náhlých změn.

Aktuální pozice kurzoru získaná z třídy Mover byla poté transformována tak aby měla souřadnicový počátek ve středu obrazu a byla odeslána do řídící aplikace. Souřadnicový střed obrazu kamery se po další transformaci na straně řídící aplikace rovnal středu projekce (kde by měla být horizontálně umístěna i kamera). Po aplikování předem definovaného multiplikátoru pak bylo možné na výsledné pozici zobrazit kurzor v projekci.

Nedostatkem této varianty implementace se ukázalo být především samotné rozpoznávání gest vzhledem k umístění kamery. Díky tomu, že kamera není umístěna ve středu projekce přímo proti uživateli ale pod projekcí aby nebránila výhledu uživatele, jsou gesta snímána z neoptimálního úhlu. Zároveň pokud uživatel současně hýbe a rotuje

rukou, úhel, pod kterým je gesto snímáno, se ještě více mění a znesnadňuje to aplikaci gesto správně rozpoznat. Tento nedostatek by nejspíše šlo odstranit přetrénováním neurálního modelu na datasetu zahrnující extrémní úhly pohledu.

Dalším nedostatkem byla absence možnosti přepnout výpočty probíhající na modelu z procesoru na grafickou kartu, což způsobilo omezení na 30 fps. Společně s předchozím problémem to vedlo k implementaci jiné varianty aplikace v jazyce Javascript s využitím knihovny TensorFlowJs a modelu Handpose [30].

Handpose model obsahoval pouze rozpoznávání pozice ruky a tím pádem jen ovládání pozice kurzoru. Další příkazy jsou závislé na hlasovém ovládání popsaném v sekci 2.3.5. Vnitřní logika, od rozpoznání ruky po odeslání pozice kurzoru do řídící aplikace, je stejná jako v předchozí variantě včetně použití třídy Mover. Díky možnosti přenesení výpočetního zatížení modelu na grafickou kartu, dosahuje model Handpose na desktopovém PC stabilních 55-60 fps. Z důvodu rychlejší interakce tak byl pro finální aplikaci upřednostněn model HandPose.

Důležitým aspektem u tohoto druhu interakce byla také nutnost přítomnosti dostatečného osvětlení, tak aby byly aplikace snadno schopny nalézt a rozpoznat ruku uživatele. To lze nejlépe za denního světla či při plném osvětlení místnosti, což v obou případech nebylo možné použít, jelikož by tím byla značně ovlivněna kvalita projekce. Z toho důvodu byly testovány různé kombinace kamer a okolního osvětlení.

Pro dosažení úplné tmy a současně fungující interakce, se testovala upravená kamera z PlayStation 3 s názvem PS3eye, která měla nahrazen filtr infračerveného světla za filtr viditelného spektra a byla tak schopna snímat obraz i za úplné tmy s infračerveným přisvícením, jež je pro lidské oko neviditelné. Taková kamera fungovala prakticky stejně jako klasická RGB kamera po přepnutí pouze do stupňů šedi. Aplikace v tomto případě byla schopna rozpoznávat statickou ruku v obraze celkem bez problémů ale po rozhýbání již přestávala zvládat dostatečně kvalitně sledovat její polohu. Obtížné se pro aplikaci při použití této kamery ukázalo i rozpoznání ruky proti většině druhů oblečení, jelikož se odstín ruky a oblečení odlišoval jen minimálně.

Jako další byla testována kamera Arducam lowlight, taktéž s odebraným filtrem infračerveného světla. S touto kamerou dosahovala aplikace již výrazně lepších výsledků i když problém s chybějícím kontrastem ruky vůči oblečení přetrvával a musel být eliminován použitím specifických druhů oblečení více pohlcujícího příchozí infračervené světlo.

Jako poslední a finální přístup byla nakonec zvolena sestava s Arducam a přisvíceným ledkovým pásem na uživatele. Tato varianta byla nakonec zvolena díky nejlepším výsledkům ve sledování pohybující se ruky a zároveň díky obavě o ovlivnění funkce EyeTracking brýlí snímajících pohyb očí uživatele v pozdějších fázi uživatelského testování. Přisvícení ledkovým pásem bylo také nastaveno tak aby co nejméně ovlivňovalo projekci na válec. Porovnání obrazů kamer z obou přístupů lze vidět na obrázku 2.13.

Závěrem je nutno říci, že použitá implementace je řešením vhodným především pro klasickou plochou obrazovku, jelikož zakřivení válce v hojné míře svádí uživatele při posouvání kurzoru k přirozenému natočení vůči aktuálně pozorované části válce. To ale kamera nereflektuje a snímá uživatele stále ze stejného úhlu, a tak může dojít k častému zmatení uživatele. Ke zmíněnému jevu dochází pak především při interakcích na stránách vzdálených od středu projekce. Tento nedostatek by bylo možno odstranit ka-

Obrázek 2.13. Vlevo obraz IR camery. Vpravo onbraz v klasickém RGB spektru.

librací obrazu kamery nebo využitím více kamer a sloučením jejich obrazu dohromady, jež však nebyli předmětem této práce.

Zároveň by se nabízelo, aby se kurzor objevoval na válci přesně tam, kam ukáže uživatel rukou či prstem. K tomu by bylo ale třeba synchronizace informace z alespoň dvou kamer, tak aby byly pokryty všechny úhly a mohl být vypočítán vektor jakým směřuje uživatelova ruka na prostor válce.

Zde implementovaná metoda je tedy nejjednodušší variantou, která neumožňuje uživateli přímo ukazovat na konkrétní místa na válci, ale nutí ho zacházet s kurzorem podobně jako při držení klasické myši. Uživatel musí primárně pozorovat kde se nachází kurzor a změnou pozice ruky pak měnit i pozici kurzoru. I přesto bylo možné splnit hlavní cíle práce a tato oblast se nabízí pro budoucí výzkum.

2.3.5 Interakce pomocí hlasu

Při implementaci aplikace pro interakci pomocí hlasu a hlasových příkazů jsem vyzkoušel obě varianty rozebírané v sekci 1.6.

Jako první byla implementována varianta v jazyce Python využívající funkce knihovny SpeechRecognition. Aplikace obsahovala hlavní smyčku, ve které vždy čekala na zaznamenání zvukového vstupu z mikrofonu, následně začala zaznamenávat přijímaný zvuk a po uplynutí časového limitu či domluvení uživatele ukončila zaznamenávání a tento záznam odeslala do specifikovaného cloudového API pro převod řeči na text. Z API byla poté zpět vrácena odpověď s výsledným přeloženým textem či chyba v případě nerozpoznání řeči. Výsledný text už byl poté na straně aplikace analyzován na základě klíčových slov a v případě nalezení shody s klíčovým slovem pro určitý příkaz byl tento příkaz odeslán do řídící aplikace. V rámci testování bylo zkoušeno pouze defaultní napojení na cloudové API Google Speech Recognition.

Tato implementace měla hned dva problémy. Díky odděleným fázím nahrávání, odesílání a rozpoznávání, se vytvářela citelná časová prodleva (1-2 vteřiny) mezi vyslovením příkazu a jeho vykonáním na řídící obrazovce. Zároveň aplikace trpěla i značnou chybovostí v rozpoznaných i nerozpoznaných slovech, která vedla k dalšímu poklesu uživatelské přívětivosti. K tomuto efektu doházelo pravděpodobně především díky velmi špatné akustice uvnitř skleněného válce, kterou nebylo možné účinně odfiltrovat, a také díky větší vzdálenosti uživatele od mikrofonu (cca 1,25 m).

Díky zmíněným nedostatkům první implementace byla implementována druhá varianta v jazyce Javascript, která využívá model SpeechCommands z knihovny TensorFlowJs. V

této variantě je vstupní hlasový příkaz okamžitě rozpoznán a nahrazen klasifikovanou pravděpodobností tříd do kterých může spadat. Třída s nejvyšší pravděpodobností, vyšší než nastavená hranice 70 procent, je vybrána jako nejpravděpodobnější a příkaz k ní přiřazený je odeslán do řídící aplikace.

Primární výhodou a zároveň limitací je možnost vytvoření a natrénování vlastních sad slov, se kterými může model pracovat. To umožňuje natrénovat model přímo na specifickou akustiku válce a zároveň zredukovat počet možností, mezi kterými se musí model rozhodovat. V protikladu k těmto výhodám to znamená omezení rozpoznávacích schopností modelu pouze na daná slova. V případě změny či přidání slov je třeba natrénovat model znovu a upravit klasifikační logiku aplikace.

Jako finální hlasové příkazy byly zvoleny tyto slova:

- **„Select"** – převedeno na příkaz uchopení objektu nad kterým se nachází kurzor
- **„Release"** – převedeno na příkaz puštění objektu
- **„Hide"** – převedeno na příkaz pro schování obrázku u obrázku obsahující i popisek
- **„Open"** – převedeno na příkaz otevření obrázku u obrázku obsahujícího popisek
- **„Click"** – převedeno na příkaz kliknutí kurzoru

Při jejich výběru bylo také třeba dbát na to, aby byla slova foneticky co nejodlišnější a předešlo se tak co nejvíce možným záměnám a chybám modelu.

Společně s aplikací pro rozpoznávání pozice ruky tak hlas doplňuje možnosti interakce, umožňující uživateli interagovat s rozhraním promítaným na skleněný válec. V případě obou aplikací implementovaných v jazyce Javascript bylo pro zprostředkování komunikace pomocí protokolu OSC potřeba navíc jednoduchého Node serveru. Ten přijímal zprávy obdržené od aplikací skrze technologii Websocket [31] a posílal je dále do řídící aplikace jako OSC zprávy odcházející po síťovém UDP.

2.3.6 Zpracování interakčních příkazů řídící aplikací

Po zpracování interakce od uživatele a odeslání příslušného příkazu do řídící aplikace, je zde příkaz přijat jako OSC zpráva a klasifikován dle své adresy. Některé z příkazů mají další argumenty s proměnnými. Aplikace v konečné implementaci umožňuje rozlišit těchto osm příkazů:

- */point/start* – přemístí kurzor na pozici danou argumenty x a y. Pozice je škálována dle počtu obrazovek.
- */point/move* – přemísti kurzor o vektor daný argumenty x a y. Vektor je zde také škálován v závislosti na počtu obrazovek.
- */drag/start* – uchop objekt. Při zpracování tohoto příkazu se prochází seznam všech objektů a porovnává se, zda se nad některým právě nenachází kurzor. Pokud ano, zkontroluje se dále zda je objekt nastaven jako uchopitelný a pokud i toto platí, tak kurzor zčervená aby indikoval uživateli změnu stavu a objekt je nastaven jako uchopený. Díky tomu dále mění svou pozici společně s kurzorem. Pokud se pod kurzorem žádný objekt nenachází, dojde jen ke krátké indikaci změnou barvy kurzoru že byl příkaz zaznamenán.
- */drag/stop* – puštění objektu. Pokud je aktuálně držen nějaký objekt, dojde k jeho uvolnění.
- */hide* – schovej obrázek. Během zpracování tohoto příkazu je proiterován seznam všech objektů a během toho je zjišťováno zda se nad nimi nenachází kurzor. Pokud ano a jedná se o obrázek s popiskem, je tento obrázek schován.

- */open* – otevři obrázek. Stejně jako v předchozím případě, jen dojde k otevření obrázku pod popiskem.
- */click* – proveď akci kliknutí nad objektem. Projde seznam všech objektů a porovná s pozicí kurzoru. Pokud se kurzor nachází nad objektem, provede na objektu operaci dle nastaveného typu ClickAction. Stejně jako u /drag/start zde dochází ke změně barvy kurzoru, tentokrát na zelenou.

Veškeré příkazy jsou přijímány a zpracovávány ve třídě **InputManagerController**. Tato třída poslouchá na přednastaveném portu (v tomto případě port 12340) a zpracovává přijaté zprávy. Interakci je možno rozšířit přidáním dalšího příkazu a implementací jeho funkcionality.

Barevná indikace stavu kurzoru je zaznamenána v proměnné *mouseState* a na jejím základě je také příslušná barva odesílána klientským aplikacím.

Kapitola 3
Uživatelské testování

Testování s uživateli je důležitou součástí této práce. Jeho cílem je otestování uživatelské přívětivosti implementovaného rozhraní, použitelnost a přívětivost interakce pomocí ruky a hlasu a také pozorování schopnosti uživatele přepínat pozornost mezi projekcí a scénou za ní. Testování proběhlo se sedmi uživateli se kterými byl vyhotoven semistruktorovaný rozhovor o uživatelově zážitku z používání rozhraní. Vyhrazený čas na test s jedním uživatelem byl 1 hodina včetně rozhovoru a uvedení.

3.1 Zvolené přístupy k testování s uživateli

Celkem bylo sestaveno pět scénářů z čehož dva z nich jsou určeny pro simulování situací blížících se reálnému použití s průhledem skrze displej a zbylé tři jsou určeny pro testování interakcí s rozhraním. První čtyři scénáře jsou popsány ve vlastní sekci níže, poslední je pak stejný jako třetí scénář, s tím rozdílem že se zde používá interakce pomocí gest místo hlasu.

Ve spojitosti s interakcemi s rozhraním bylo cílem sledovat, jaký vliv má na každého uživatele rozpor mezi vnímáním zakřivení válce a plochým snímáním rukou uživatele jednou kamerou. Hypotéza byla, že minimálně z počátku bude tento rozpor způsobovat u uživatelů jisté zmatení a v některých případech bude vyžadovat i zásah návodnou instrukcí, jak se dostat dále.

Proto abychom mohli sledovat uživatelovu pozornost, měl během testů na sobě brýle pro sledování pohybu očí (dále jen eye tracking) Tobii pro glasses 2 od firmy Tobii [32]. Tyto brýle, viditelné na obrázku 3.1, umožňují, po kalibraci na konkrétního uživatele, sledovat oči uživatele a v kombinaci s přední kamerou brýlí vyhodnocovat, kam se uživatel právě dívá. Z kombinace pohledu kamery brýlí a sledované pozornosti uživatele je poté vytvářen videozáznam, který slouží k dalšímu zpracování.

Obrázek 3.1. Eye tracking brýle od firmy Tobii (převzato z [33]).

V rámci příprav testování bylo také nutné upravit samotnou aplikaci a dodat do ní nejenom všechny potřebné prvky pro níže zmiňované scénáře, ale také možnost simulovat všechny hlasové příkazy klávesami z řídící aplikace, tak aby se v případě úplného selhání hlasového ovládání dalo toto ovládání nahradit metodou Wizard of Oz.

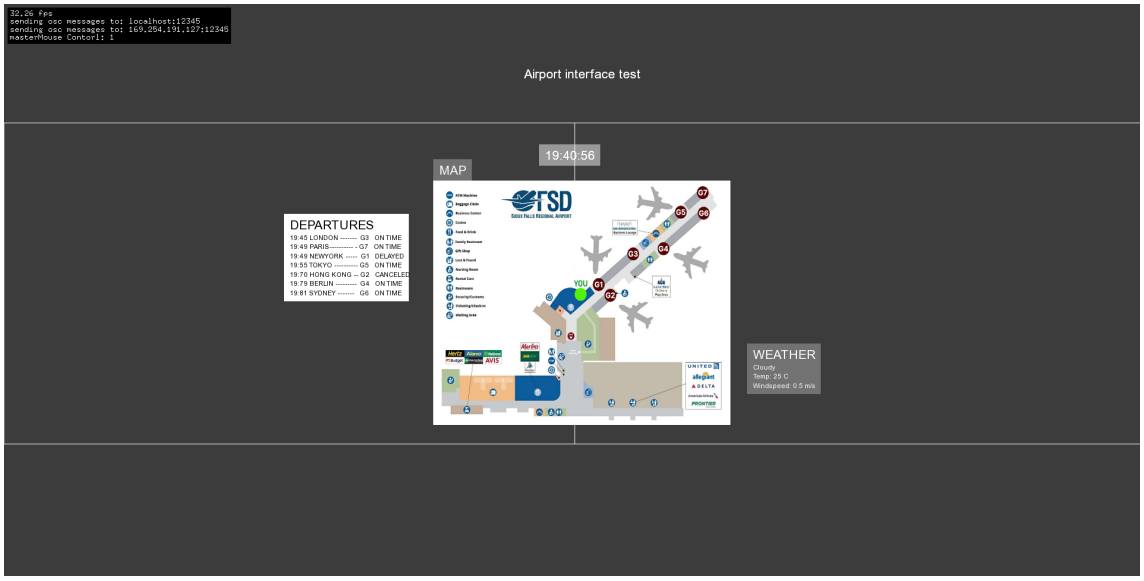
Před každým scénářem byl uživatel proškolen o způsobech ovládání. V rámci uvedení do nového stylu ovládání bylo důležité dát možnost uživateli si toto ovládání nanečisto vyzkoušet abychom eliminovali učící efekt.

3.1.1 Letištní informační terminál

Cílem tohoto scénáře je prvotní seznámení uživatele s rozhraním, vyzkoušení si orientace v rozhraní spolu s interakcemi a úvodní vybudování představy o průhledném displeji, později dále rozebírané v následném rozhovoru. V rámci toho měl uživatel představovat cestujícího na letišti, který hledá aktuální informace o svém letu a zároveň odletovou bránu, ze které mu letadlo odlétá. Tyto informace hledá na informačním rozhraní zabudovaném přímo do jednoho z oken terminálu, ze kterého je možný výhled na letiště. Kroky uživatele by měli být tedy následující:

- Vyhledání správného letu
- Zjištění času odletu a odletové brány ze které letadlo odlétá
- Otevření mapy a nalezení odletové brány
- Vyrazit směrem k odletové bráně

V rámci této části nebyl uživatel limitován časem, a měl možnost se volně seznámit s rozhraním a splnit zadaný úkol. Zároveň tato část ani příliš necílila na sledování uživatelovi pozornosti mezi reálnou scénou a rozhraním, jelikož zde uživatel nemá žádný scénářem daný úkol sledovat během interakce s rozhraním dění za ním. Pro představu lze na obrázku 3.2 vidět snímek obrazovky rozhraní z řídící aplikace.



Obrázek 3.2. Ukázka obrazovky řídící aplikace prři rozestavení objektů pro scénář Letiště.

3.1.2 Sledování chodců

V tomto scénáři jde o sledování simulace reálné scény na stěně místnosti za rozhraním a porovnávání sledovaného obsahu s obsahem zobrazovaným na rozhraní průhledného

displeje. Uživatel představuje pracovníka bezpečnosti hledajícího v procházejících chodcích nebezpečné osoby. Tyto osoby má zobrazené na průhledném displeji, kde má dále také viditelné informace o aktuálním místě, na kterém se nachází. V ideálním případě by tento scénář měl simulovat pracovníka bezpečnosti či vrátného na vrátnici větší instituce, kteří bývají často za zasklenou přepážkou, kterou by mohlo být možné jako průhledný displej použít. Jako náhrada za nahrávku z takového místa bylo pro reálnou scénu zvoleno sestříhané video lidí procházejících po ulici ve smyšlených lokalitách. Názvy těchto lokalit se uživateli vždy zobrazují v informačním okně na průhledném displeji a jelikož není vytvořena žádná automatická synchronizace mezi videem a scénou, jsou měněny manuálně.

Úkolem uživatele je sledovat video procházejících lidí na reálné scéně a snažit se nalézt hledané osoby, které bude mít zobrazené na válci. Pokud takovou osobu nalezne, měl by nahlásit lokalitu, v níž se nachází.

Ostrému scénáři předchází vzorové kolo s hledáním jedné osoby, na které si uživatel úkol vklidu vyzkouší. Osoba i video v tomto případě budou odlišné od těch použitých v ostrém testu. Ukázku připravené obrazovky z ostrého testu lze vidět na obrázku 3.3.

Obrázek 3.3. Ukázka obrazovky řídící aplikace pro scénář Sledování chodců (Security).

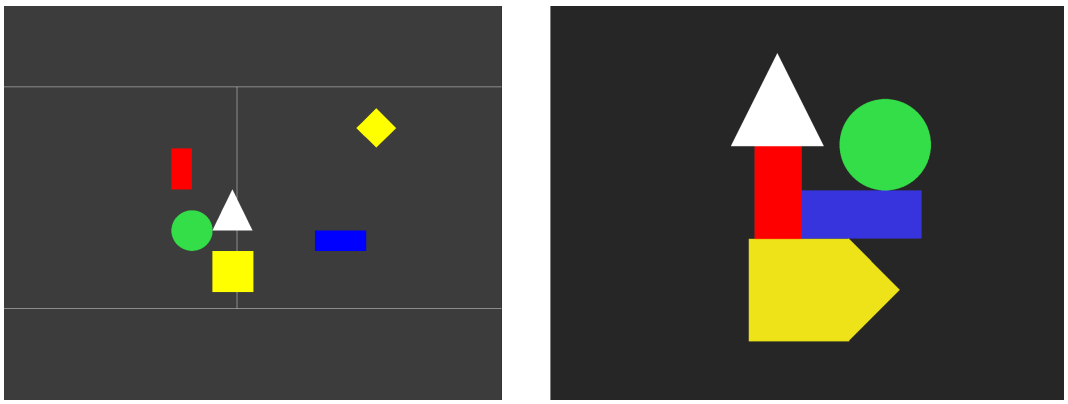
3.1.3 Skládání obrazce

Jedná se o jednoduchý, čistě umělý scénář sloužící především k dalšímu prohloubení uživatelovi interakce s rozhraním pomocí rukou a hlasu. Uživatel má v tomto scénáři za úkol sestavit obrazce na průhledném displeji do podoby zobrazené na nástěnné projekci za ním. Lze tu tedy sledovat zároveň i určitou míru přesouvání pozornosti mezi jednotlivými vrstvami, předpoklad je ale takový, že tento jev zde nebude příliš výrazný, jelikož lze počítat s tím, že si uživatel zvládne zapamatovat sestavu jen letmým prohlédnutím. To o co tedy v tomto případě jde především, je rychlost a přesnost, s jakou zvládne uživatel danou sestavu složit. Ukázku sestavy a iniciálního rozložení z řídící aplikace lze vidět na obrázku 3.4.

Hypotézou pro tento scénář, provedený pro oba druhy interakce (pomocí hlasu i gest),

bylo, že uživatel by měl být rychlejší a přesnější ve skládání zadaného tvaru pomocí gest. Jelikož reakce na gesta je o něco rychlejší a vyžaduje zapojení stále stejných kognitivních funkcí.

Získaná uživatelova zkušenost z této části experimentu by měla být dále rozebrána v semistrukturovaném rozhovoru.



Obrázek 3.4. Vlevo iniciální rozložení obrazců na průhledném displeji. Vpravo požadovaná sestava.

3.1.4 Fitt's law experiment pomocí rukou

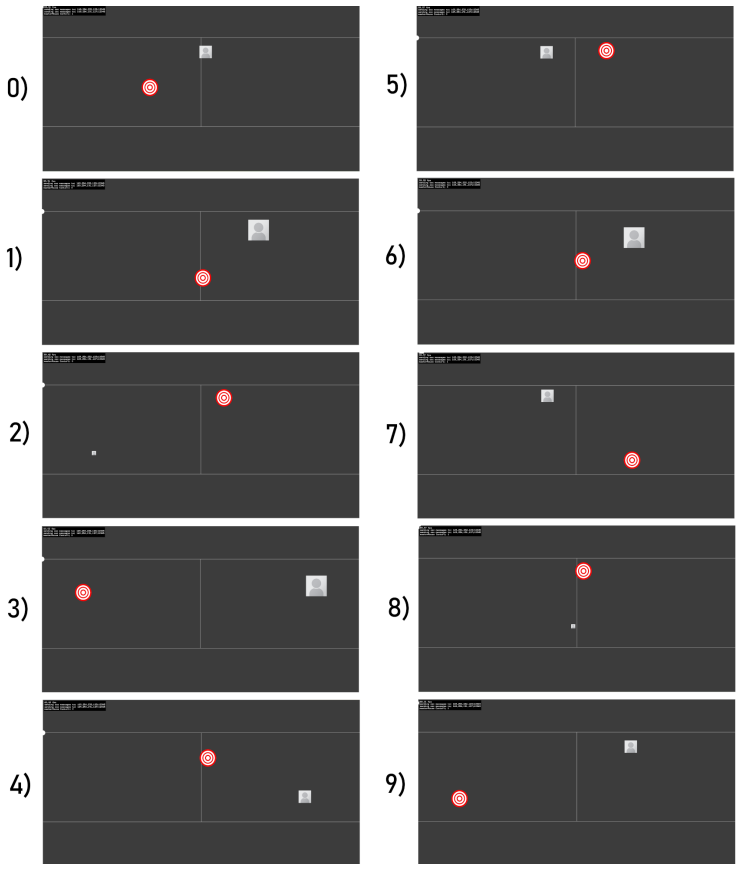
Tento scénář se zakládá na myšlence Fittsova zákona (Fitt's law)[34], jenž nám udává vztah mezi rychlostí jakou jsme schopni se ze startovního bodu dostat nad cílový objekt, v závislosti na vzdálenosti a velikosti tohoto objektu. Cílem scénáře je prozkoumat, jak rychle uživatel zvládá manipulovat s objekty v rozhraní v závislosti na jejich vzdálenosti od sebe, umístění na obrazovce a na jejich velikosti.

Pro účel scénáře byl sestaven jeden vzorový a deset testovacích příkladů, ve kterých je vždy rozmístěn objekt o variabilních rozměrech a cíl do kterého má být objekt přesunut. Rozměry objektů jsou stanoveny na tři základní úrovně. Nejmenší má 50px, což na válci znamená cca 10 cm, prostřední pak 150px a největší 250px, což na válci znamená cca 50 cm. Vzdálenosti a umístění objektů a cílů jsou mezi příklady dány náhodně po celé ploše používané obrazovky válce. Jednotlivé rozložení v testovacích příkladech lze vidět na obrázku 3.5.

Pro každý testovaný příklad se počítá čas od doby, kdy uživatel uchopí objekt, do doby puštění objektu nad cílem. Za tímto účelem se při každé akci zapisuje aktuální čas a v případě puštění objektu i jeho výsledná pozice. Výsledný zápis ze všech příkladů je pak převeden do tabulky. Ze získaných výsledků měření by poté mělo být možné pozorovat vliv velikosti objektů (a případně jejich vzdálenosti) na dobu plnění úkolu. Zároveň by měl být také patrný určitý vliv velikosti objektů na chybu, danou vzdáleností mezi cílem a uživatelem finálně umístěným objektem.

3.1.5 Semistrukturovaný rozhovor

Účelem tohoto rozhovoru bylo zmapování uživatelova zážitku po projití všech testovacích scénářů, zjištění jeho pocitů a poznatků z používání interakcí s rozhraním a v poslední řadě také probrání proběhlé zkušenosti s používáním průhledného displeje. Návodné otázky zvolené pro vedení rozhovoru jsou popsány v příloze A.



Obrázek 3.5. Obrázek s rozložením všech deseti testovacích příkladů pro experiment s Fittsovým zákonem. Červeně je znázorněn cíl a šedě pohyblivý objekt.

3.2 Výsledky testování

V konečném výsledku proběhlo testování se sedmi uživateli. Jednalo se o dvě ženy a pět můžu. Věkově byla nejvíce zastoupena skupina 21-24 let a poté bylo zbylé věkové zastoupení v orientačním rozmezí 30-45 let. Většina uživatelů byla studiem či prací vztahově spjata s IT. 3 z uživatelů dokonce přímo UX, a zastoupeno zde bylo například i medicínské odvětví.

Testování proběhlo, až na jeden případ, během jednoho dne v plánovaných hodinových úsecích, které byly akorát dostačující pro uvedení do experimentu, průchod všemi scénáři, závěrečný rozhovor a rychlou přípravu na dalšího uživatele. Pro identický průběh všech scénářů byl sestaven také protokol testování, který je přiložen k práci v příloze A. Níže jsou v samostatných sekcích jednotlivě rozebrány cílené aspekty použitelnosti aplikace a průhledného displeje.

3.2.1 Vyhodnocení sledování pozornosti uživatele

Pozornost uživatelů byla sledována především pomocí eyetracking brýlí, u kterých byl následně vyhodnocován záznam pořízený brýlemi během experimentu. Přesto že se brýle podařilo počátečně úspěšně kalibrovat pro všechny uživatele, v průběhu se u velkého množství uživatelů objevilo porušení záznamu uživatelovi pozornosti způsobené chybou či úplnou ztrátou ve sledování očí. Jak bylo zmíněno v předchozí sekci, pozornost uživatele byla sledována především u scénáře Sledování chodců a v menší míře pak také u scénáře Letiště. Výsledkem testování obou z scénářů byla stoprocentní a celkově

bezproblémová úspěšnost v plnění zadaného úkolu v případě scénáře Letiště a průměrně 60% úspěšnost u scénáře Sledování chodců, kde se dařilo uživatelům najít průměrně 3 z 5 hledaných osob.

Ve spojitosti s pozorností uživatele bylo zaznamenáno značné omezení ve schopnosti střídat pozornost mezi oběma vrstvami během interakce s válcem. Jelikož byla interakce často náročná a nepřesná, vyžadovala uživatelovu plnou pozornost, což vedlo k omezení schopnosti současně sledovat druhou (reálnou) vrstvu.

Dále bylo pozorováno, a později ověřeno během rozhovorů, že schopnost v překryvu pozorovat obě aktivní vrstvy byla narušena malým kontrastem mezi objekty na skleněném válci a simulovanou scénou za nimi. Uživatel tak musel často měnit úhel pohledu tak aby měl v pozadí za objektem na válci tmavší, jednolitější pozadí, na kterém by mohl obsah objektu rozeznat. V tom důsledku uživatelé také preferovali umístění objektů vizuálně na okraje simulované projekce, což je poté vzhledem k velkým rozměrům válce, nutilo k pohybu hlavou a přesouvání plné pozornosti ze scény na válec a naopak.

Někteří uživatelé v rozhovorech také zmiňovali potřebu poodstoupení od stěny válce (to nebylo v rámci experimetnu umožněno) pro to, aby mohli mít obě obrazovky v jednom zorném poli a nebyla nutná změna akomodace oka při přesouvání pozornosti mezi vrstvami.

Výhody v kognitivním oddělení obou vrstev nebyly díky výše zmiňovaným problémům jednoznačně potvrzeny a ani uživatelé na tento aspekt neměli jednotný názor. Do budoucna je tak třeba tyto problémy eliminovat zvýšením kontrastu obou vrstev a lepším pozicováním uživatele.

3.2.2 Vyhodnocení interakce s rozhraním

V rámci interakce uživatelů s rozhraním byly pozorovány zřetelné rozdíly mezi jednotlivými uživateli. Část z nich se zvládala okamžitě přizpůsobit novému typu interakce a pochopit jeho logiku, většina ale s interakcí po celou dobu různými způsoby bojovala a nejednalo se tak pro ně úplně o příjemnou uživatelskou zkušenost. Konkrétní problémy jsou rozebrány na konci této podsekce.

V případě scénářů Skládání obrazce, jež byly určeny k testování přesné manipulace s objekty v rozhraní, byl především u hlasového ovládání pozorován velký vliv zpoždění mezi vyslovením příkazu a jeho provedením v aplikaci. To způsobovalo nepříjemnou nutnost kalkulovat dopředu kdy vyslovit příkaz tak aby objekt skončil tam kde uživatel chce. Zároveň tato prodleva také nutila uživatele říct příkaz a současně s tím držet stále stejnou přesnou pozici, což bylo pro mnoho z uživatelů velmi náročné. Do budoucna je tedy třeba snížit latenci dalším zrychlením interakčních aplikací a jejich komunikace s řídící aplikací.

U ovládání gesty byly tyto problémy sice eliminovány a ovládání mělo rychlejší odezvu a větší plynulost, ale objevoval se zde u některých uživatelů problém s chybným rozpoznáváním gesta otevřené ruky. Aplikace gesto u těchto uživatelů často zaměňovala za gesto OK, což vedlo ke ztrátě pohybu kurzoru a způsobovalo to ztracení uživatele. Dalším problémem u této verze byl také jemný pohyb při finálním pohybování s objekty, kdy docházelo při delším setrvání na jednom místě k rozkmitání kurzoru a rozhození přesného pozicování. Jednalo se pravděpodobně o neošetřenou chybu modelu, jelikož v daný moment začalo kmitat celé rozpoznávání ruky.

Metriky získané při těchto scénářích bohužel nejsou nijak průkazné, jelikož jsou často neúplné a zahlcené vlivem úrovně uživatelovi únavy a poklesu zájmu. To se reálně

odráželo v obecně méně přesném rozvržení objektů u druhé varianty s gesty, jelikož uživatelé již byli unavení a neměli takovou motivaci trávit na scénáři více času. I přesto jsou získané naměřené časy vypsané v tabulce na konci přílohy C. Zde lze vidět, že průměrný čas byl delší u ovládání gesty, ale medián byl naopak o cca 20 sekund kratší než v případě ovládání hlasem. Medián je v tomto případě průkaznější díky robustnosti vůči velkému vlivu jednoho vychýleného vzorku měření, který se projevuje u průměru.

U Fitt's law experimentu došlo k špatnému uchopení experimentu a v souvislosti s tím k chybě při jeho iniciálním nastavení. Oproti myšlence Fittsova zákona, kde je pozorován vztah času potřebného pro dosažení cíle ku rychlosti (se kterou se uživatel pohybuje k cíli) a velikosti cíle, byla v tomto případě měněna velikost posouvaného objektu a ne cíle. Z toho důvodu nelze získaná měření přímo použít k vyhodnocení Fittsova zákona, ale k vytvoření vlastních nezávislých závěrů. Výsledky zmiňovaných měření lze vidět v tabulce 3.1.

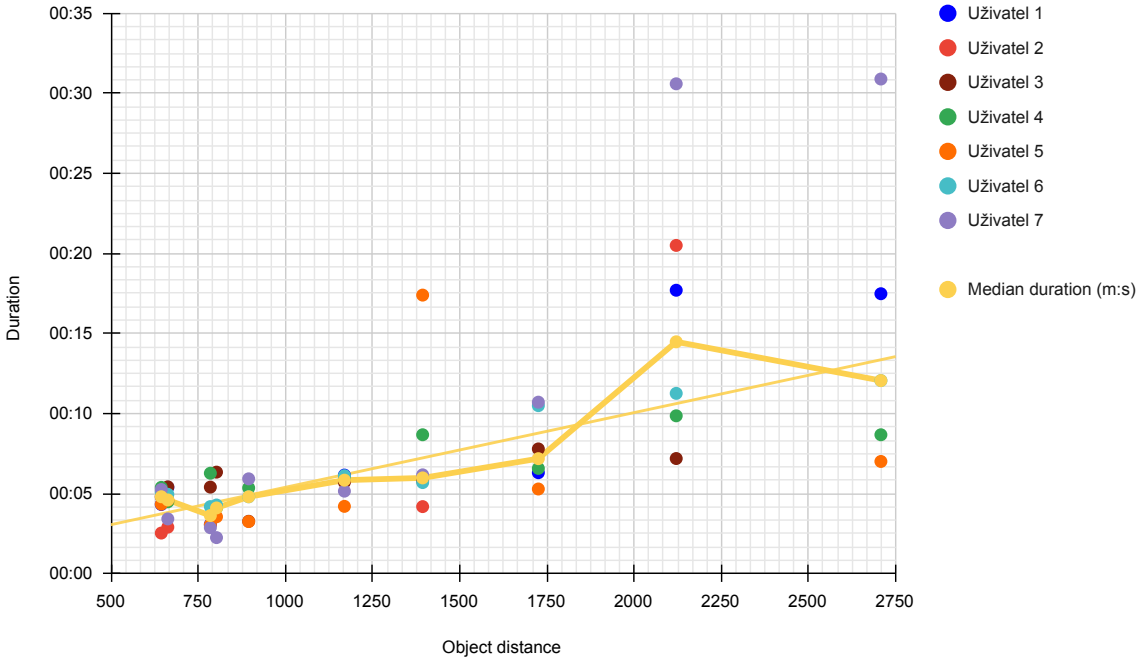
Úkol	Start X	Start Y	Size	Cíl X	Cíl Y	Vzdálenost	Průměrný čas	Medián času	Průměrná chyba
0	1900	100	150	1300	605	784	00:04	00:04	70
1	2500	100	250	1950	805	894	00:05	00:05	79
2	600	800	50	2200	155	1725	00:08	00:07	34
3	3200	200	250	500	405	2708	00:15	00:12	91
4	3100	700	150	2000	305	1169	00:06	00:06	111
5	1500	100	150	2300	155	802	00:04	00:04	73
6	2500	200	250	2000	605	643	00:05	00:05	65
7	1500	50	150	2600	905	1393	00:08	00:06	60
8	1850	800	50	2000	155	662	00:04	00:05	57
9	2500	100	150	500	805	2121	00:16	00:14	51

Tabulka 3.1. Tabulka s výsledky měření ve vztahu k počátečním podmínkám úkolů. Start X/Y – počáteční pozice objektu, Cíl X/Y – pozice cíle, Vzdálenost – úvodní vzdálenost objekt-cíl. Jednotky vzdálenosti jsou v pixelech.

Z výsledků v tabulce lze pozorovat předpokládaný vliv vzdálenosti mezi objektem a cílem na výsledný čas potřebný pro přesunutí objektu nad cíl. Celkem logicky lze zřetelně vidět že vyšší vzdálenost znamená obecně delší čas potřebný pro přesunutí objektu. U úkolů 3 a 9, které mají nejdelší vzdálenost mezi objektem a cílem, lze vidět že průměrný čas pro splnění úkolu 3 (jež má celkově největší vzdálenost) je nižší než u úkolu 9. To lze přisuzovat velikosti objektu, která je v tomto případě nejvyšších 250 px.

Při pohledu na graf 3.6, vynesený ze všech měřených časů, lze s rostoucí vzdáleností mezi cílem a objektem dále pozorovat zvětšující se rozptyl mezi časy jednotlivých uživatelů potřebných pro dokončení úkolu. Při zpětném prohlížení záznamů to bylo částečně přisouzeno chybám ve sledování rukou, které byly často způsobeny příliš rychlím pohybem ruky uživatele, kterou již model nestíhal sledovat. Velký vliv měla ale také rozdílná úroveň v intuici k ovládání u jednotlivých uživatelů, kdy některým nedělalo problém prohození rukou během uchopení, jiní se pak snažili o přetáčení, aby dosáhli na pozici cíle stejnou rukou, nebo v mezičase pokládali objekt a poté jej teprve uchopovali tou druhou. Pro další experimenty je tedy třeba kontrolovat předchozí znalost uživatelů například delším testováním nanečisto.

Další poznatek byl získán přímo v průběhu testování a později i z pořízených videozáznamů. Při úvodním uchopování objektů uživateli, byly u všech znatelné problémy v uchopení objektů nacházejících se na pravém či levém okraji projekce. To bylo jasně způsobeno rozdílným stylem interakce, a intuice uživatele. Uživatelé často předpoklá-



Obrázek 3.6. Graf zobrazující časy potřebné pro dokončení jednotlivých úkolů v závislosti na vzdálenosti objektu a cíle.

dali prostorovost interakce a tak se často natáčeli vůči stranám ve snaze chovat se stejně jako při otočení do středu projekce. Interakční aplikace je ale stále snímala ze stejného místa ve středu projekce, a tak dávala uživateli úplně jiné výsledky než by očekával.

S uchopováním objektů se také dále pojí jejich velikost, kde se jako již příliš malá ukazovala nejmenší velikost 50px (tedy cca 10 cm), která byla pro uchopení významně náročnější.

Z obecných pozorování během testů bylo zjištěno absolutní selhání hlasového rozpoznávání u ženských uživatelů. V jejich případě mělo hlasové rozpoznávání příkazů zhruba patnáctiprocentní úspěšnost a vyšší byla jen v případě, že se uživatelka snažila modulovat hlas do hlubšího tónu či zvýraznila určité části slov. Tato chybovost mohla být do jisté míry způsobena doučováním modelu jen na vzorcích mužského hlasu. I u mužských uživatelů se ale často stávalo, že příkaz byl rozpoznán třeba až na podruhé, a tak nelze považovat hlasové ovládání za spolehlivé. Je tedy patrný genderový bias, který je nutné pro budoucí vývoj odstranit a obecně dále zrobustnit hlasovou interakci.

Dalším faktorem posilujícím nevhodnost hlasového ovládání byl také fakt, že uživatel nemohl během interagování s rozhraním mluvit, jelikož to způsobovalo chybné zaznamenání a provedení některých příkazů které uživatel nechtěl. V reálném použití lze předpokládat, že by většina uživatelů ocenila možnost během takové základní interakce mluvit a toto omezení by jim bylo velmi nepříjemné.

Poznatkem, směřujícím ke komponentám zobrazovaného uživatelského rozhraní, byl častý omyl uživatelů při snaze otevřít pomocí příkazu open i komponenty obsahující pouze statické informace. Zde by bylo vhodné doplnění určité indikace že lze danou komponentu otevřít či zavřít.

3.2.3 Dodatečné poznatky z rozhovorů

Většina z poznatků získaných během rozhovorů byla již zmíněna v předchozích částech. Zde tedy zbývá prostor už jen pro shrnutí těch zbývajících.

Jako nejčastější faktor, spojující většinu uživatelů, se ukázalo být značné vyčerpání rukou po skončení celého experimentu. Většina uživatelů měla po celou dobu interakce s displejem plně nataženou ruku, což způsobovalo, společně s dlouhou dobou experimentu, o to výraznější vysílení. U menšiny uživatelů, kteří se naučili pracovat s rozhraním s pokrčenou rukou, pak bylo toto vysílení znatelně menší. Tento fenomén se nazývá gorilla arm syndrom[35] a ukazuje limity rozhraní s gesty, Gesta je třeba adekvátně přizpusobit aby se tento syndrom co nejvíce omezil.

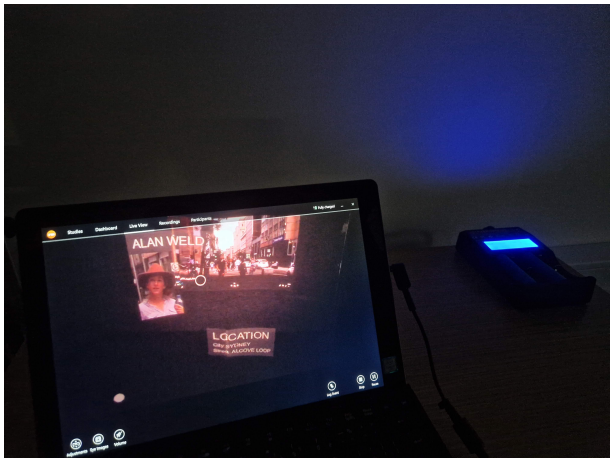
Společně s fyzickým vysílením se také často objevovalo nezanedbatelné vysílení z kognitivní náročnosti celého experimentu, spojené s náročností interakce s rozhraním. Ta byla posílena již dříve zmiňovanou prodlevou ve vykonání hlasových příkazů, komplikovanějším ovládáním kurzoru, ale také častými chybami.

Při probírání možností reálného využití průhledného displeje i podobného druhu interakce zaznívalo nejčastěji spojení s informačními panely v obchodních centrech. Zde by nebylo pravděpodobně možné ani potřebné nijak přímo využít vlastnost průhlednosti displeje, spíše jen jako estetický bonus pro přitáhnutí pozornosti. Reálné užití interakce zde však zmiňovali hned dva uživatelé. Důvodem k tomu byla hygienická výhoda vzdálených interakcí používaných v této práci, oproti přímému dotyku na displeji. Tento hygienický nedostatek u současných veřejných dotykových obrazovek znechucoval uživatele především v době covidové pandemie.

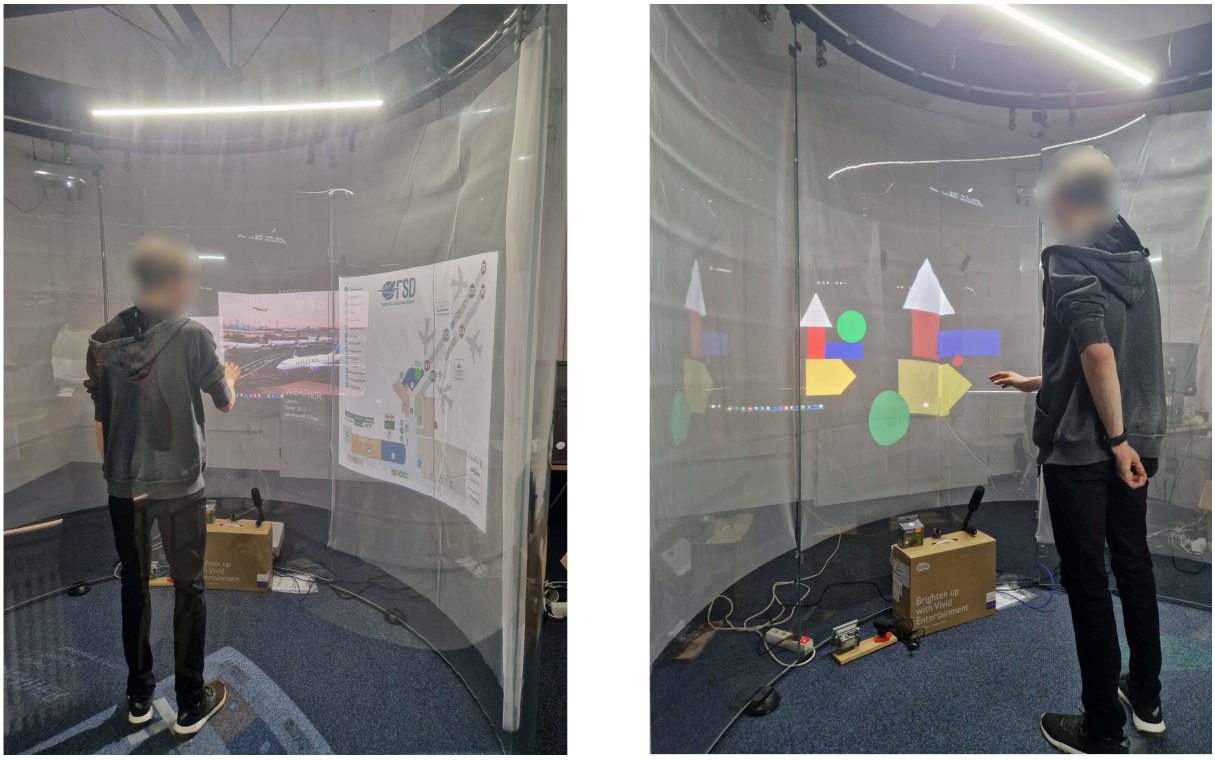
Jako další možnost reálného užití průhledného displeje byla zmiňována komerční akvária, kde by mohl být displej integrován přímo do skleněné stěny a mohl by pomáhat zobrazovat uživateli informace o proplouvajících rybách. Další variantou bylo pak zobrazování informací o zastávkách či okolních památkách na oknech prostředků veřejné dopravy.

Celé poznáky z jednotlivých rozhovorů lze nalézt v příloze B.

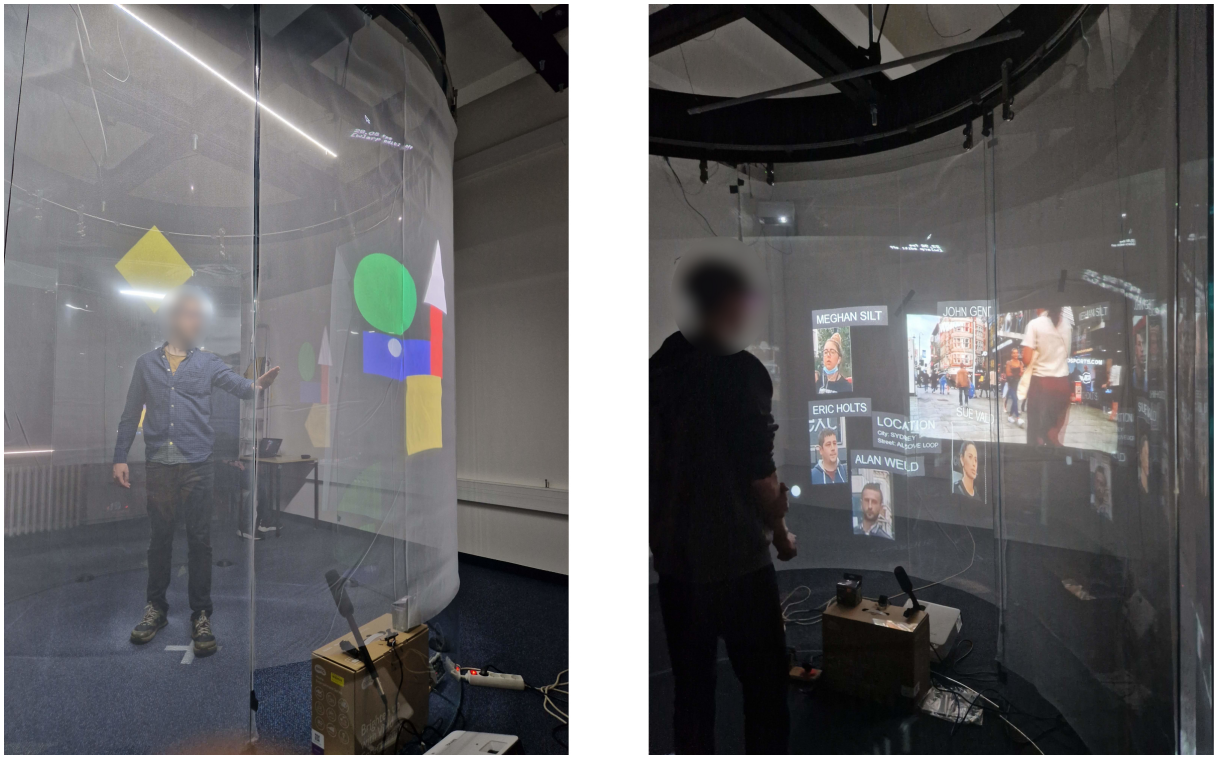
3.2.4 Fotodokumentace z průběhu testování



Obrázek 3.7. Fotografie znázorňující aplikaci pro ovládání brýlí s vyznačeným centrem kam se uživatel právě dívá (žluté kolečko).



Obrázek 3.8. Vlevo uživatel interagující s displejem v průběhu scénáře Letiště. Vpravo přesné pozicování objektů během scénáře Skládání obrazce.



Obrázek 3.9. Vlevo fotografie uživatele z druhé strany průhledného displeje. Vpravo fotografie rozhraní a uživatele během scénáře Sledování chodců.

Kapitola 4
Závěr

Během diplomové práce se podařilo vytvořit ucelenou, snadno rozšiřitelnou hardwarovou sestavu pro projekci na skleněný válec se dvěma instalovanými projektory včetně řízení vypínání a zapínání projektorů z řídícího počítače. Dále byla úspěšná implementace klientské aplikace umožňující manuálně nastavit mapování projekce na válec a zobrazovat obsah dle instrukcí zasílaných skrze protokol OSC. Zároveň také implementace řídící aplikace synchronizující obraz jednotlivých projektorů do podoby jednolité obrazovky s libovolnou škálovatelností počtu projektorů a umožňující vytváření vlastního základního uživatelského rozhraní i jeho následné ovládání, opět skrze protokol OSC. V poslední řadě byly vytvořeny a porovnávány rozdílné verze interakčních aplikací s využitím hlasu, společně se sledováním ruky a jejích gest pomocí RGB kamery. Obě tyto aplikace byly uzpůsobeny pro komunikování svých výstupů skrze OSC protokol do libovolné další aplikace.

Součástí práce bylo také testování se sedmi uživateli, jehož cílem bylo sledovat pozornost uživatelů při používání průhledného displeje a otestování použitelnosti a uživatelské přívětivosti interakcí společně s uživatelským rozhraním jako celku. Výsledkem tohoto testování byly zjištěné přetrvávající nedostatky v interakčních aplikacích v podobě velkého zpoždění reakce na uživatelem daný příkaz, částečně chybové rozpoznávání gest a také přizpůsobení ovládání kurzoru vhodné spíše pro ploché obrazovky, a ne pro takovouto prostorovou projekci uvnitř válce. Jako viditelný bonus této interakce vnímala část uživatelů její celkovou hygieničnost ve vztahu k informačním panelům ve veřejném prostranství a srovnáním s hojně rozšířenou dotykovou interakcí.

Hlasové ovládání se v podobě, v jaké zde bylo použito, ukázalo jako příliš chybové a zároveň ve větším měřítku nepříliš uživatelsky přívětivé. V ideální aplikaci by bylo vhodnější obstarání základních interakcí kompletně pomocí gest a přenechání hlasových příkazů pouze pro komplexní příkazy řešené podobným aktivačním stylem jako tomu je například u současných hlasových asistentů, poslouchajících po vyslovení specifického spouštěcího slova.

Sledování pozornosti uživatele bylo prováděno za pomoci eyetracking brýlí sledujících kam se uživatel právě dívá. Během pozorování se ukázalo, že kontrast projekce na válec a scény zobrazované za ním je příliš malý na to, aby bylo bez problému možné rozeznat obě vrstvy v kompletním překryvu, a také, že by pro eliminaci změny akomodace oka při přesouvání pozornosti mezi vrstvami bylo třeba pravděpodobně většího odstupu od stěny válce.

V rámci možného budoucího navázání na tuto práci, by bylo možné pokračovat několika směry. Jako nejrelevantnější se mi jeví využít současný systém projekce a myšlenku interakcí, doplnit projekci na válec na kompletních 360° a pracovat s myšlenkou virtuálního rozhraní, ovládaného prostorově pomocí gest. Taková instalace by mohla být vhodná především pro prostorové zobrazování různých 3D modelů, map streetview i k obecné vizualizaci velkého množství dat. Jako odlišná cesta se pak nabízí další rozpra-

cování interakcí gesty s plochou obrazovkou, určeného například právě pro informační plochy ve veřejných prostorách. V poslední řadě lze také dále pracovat s myšlenkou průhledného displeje a snažit se dále zlepšit úroveň kvality projekce na sklo včetně dalšího navazujícího výzkumu uživatelovi pozornosti.

Na závěr bych rád poděkoval svému vedoucímu práce MgA. Vojtěchu Leischnerovi za podporu během celého procesu a pomocnou radu či nasměrování správným směrem v časech kdy jsem tápal. Speciální dík náleží také Ing. Miroslavu Laco, PhD. z Fakulty informatiky a informačních technologií v Bratislavě, jež s námi během testování spolupracoval a pomáhal nám technicky a odborně se vším kolem eyetracking brýlí, které nám ze své fakulty rovněž zapůjčil.

Literatura

[1] Automotive head-up display. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2023 [cit. 2023-05-05]. Dostupné z: https://en.wikipedia.org/wiki/Automotive_head-up_display

[2] Hudway: automotive HUDs. In: *Hudway* [online]. Huntington Beach, CA., 2023 [cit. 2023-05-12]. Dostupné z: https://hudway.co/drive

[3] Rear Projection Films. In: *PRO films: Rear projection films* [online]. PRO films, 2023 [cit. 2023-02-12]. Dostupné z: https://www.rearprojectionfilms.com/product/rear-projection-films/

[4] BOUCHARD, David. *Keystone: Processing projection library* [online]. 2013 [cit. 2023-01-13]. Dostupné z: https://keystonep5.sourceforge.net

[5] O'CONNOR, Taylor. SketchMapper. In: *Github* [online]. 2018 [cit. 2023-02-13]. Dostupné z: http://josephtaylor.github.io/sketch-mapper/

[6] DEBOISBLANC, Jenna. P5.mapper. In: *Github.com* [online]. 2023 [cit. 2023-01--13]. Dostupné z: https://github.com/jdeboi/p5.mapper

[7] *P5js: Processing for javascript* [online]. Processing Foundation [cit. 2023-01-14]. Dostupné z: https://p5js.org

[8] *Processing* [online]. Processing Foundation [cit. 2023-01-14]. Dostupné z: https://processing.org

[9] OfxWarp: warp projection library. In: *Github.com* [online]. 2018 [cit. 2023-01-14]. Dostupné z: https://github.com/prisonerjohn/ofxWarp

[10] RIJNIEKS, Krisjanis. *PiMapper - projection mapping app for RaspberryPi* [online]. In: . 2018 [cit. 2023-01-14]. Dostupné z: https://github.com/kr15h/ofxPiMapper

[11] *OpenFrameworks: open source C++ toolkit for creative coding* [online]. [cit. 2023--01-14]. Dostupné z: https://openframeworks.cc

[12] DIPIETRO, L., A.M. SABATINI a P. DARIO. A Survey of Glove-Based Systems and Their Applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* [online]. 2008, 38(4) [cit. 2023-04-14]. ISSN 1094-6977. Dostupné z: doi:10.1109/TSMCC.2008.923862

[13] Gesture-based remote-control system using coordinate features - Scientific Figure on ResearchGate. In: *ResearchGate* [online]. c2008-2023 [cit. 2023-04-14]. Dostupné z: https://www.researchgate.net/figure/Figure-1-Data-glove-with-flex-sensors-Ghotkar-et-al-2012_fig1_303127129

[14] OUDAH, Munir, Ali AL-NAJI a Javaan CHAHL. Hand Gesture Recognition Based on Computer Vision: A Review of Techniques. *Journal of Imaging* [online]. 2020, 6(8) [cit. 2023-02-14]. ISSN 2313-433X. Dostupné z: doi:10.3390/jimaging6080073

[15] BAZAREVSKY, Valentin, Valentin GRISHCHENKO, Ivan GABRIEL BAZAVAN, et al. *On-device Real-time Hand Gesture Recognitio*. [online]. 2021 [cit. 2023--02-15]. Dostupné z: https://arxiv.org/abs/2111.00038

[16] MediaPipe: Hand landmarks detection guide. In: *Google Developers* [online]. 2023--05-09 [cit. 2023-05-14]. Dostupné z: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

[17] BAZAREVSKY, Valentin, Valentin GRISHCHENKO, Ivan GABRIEL BAZAVAN, et al. *3D Hand Pose with MediaPipe and TensorFlow.js* [online]. 2021 [cit. 2023-02-14]. Dostupné z: https://blog.tensorflow.org/2021/11/3D-handpose.html

[18] *MediaPipe: hand gestures*. In: *Google Developers*. [online] [cit. 2023-02-14] https://developers.google.com/mediapipe/solutions/vision/gesture_recognizer

[19] *HandPose: ml5 library* [online] [cit. 2023-02-14] Dostupné z: https://learn.ml5js.org/#/reference/handpose

[20] *Google Assistant: How Google Assistant helps you get things done* [online]. Google for developers [cit. 2023-02-14]. Dostupné z: https://developers.google.com/assistant/howassistantworks/responses

[21] Short-time Fourier transform. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2022 [cit. 2023-05-14]. Dostupné z: https://en.wikipedia.org/wiki/Short-time_Fourier_transform

[22] Simple audio recognition: Recognizing keywords. *TensorFlow* [online]. 2023-01-14 [cit. 2023-05-14]. Dostupné z: https://www.tensorflow.org/tutorials/audio/simple_audio

[23] *Teachable machine* [online]. Google [cit. 2023-02-14]. Dostupné z: https://teachablemachine.withgoogle.com

[24] SpeechRecognition: Python library. In: *Github.com* [online]. 2023 [cit. 2023-05-14]. Dostupné z: https://github.com/Uberi/speech_recognition#readme

[25] Framebuffer. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2023 [cit. 2023-04-14]. Dostupné z: https://en.wikipedia.org/wiki/Framebuffer

[26] JONGEJAN, Jonas, LINDEGAARD, Bichel Johan. OfxStreamer: image streaming library. In: *Github.com* [online]. 2014 [cit. 2023-01-14]. Dostupné z: https://github.com/HalfdanJ/ofxStreamer

[27] Open sound control. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, poslední úprava 1.1.2023 [cit. 2023-05-14]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Open_Sound_Control&oldid=1130824556

[28] User Datagram Protocol. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2023 [cit. 2023-04-14]. Dostupné z: https://en.wikipedia.org/wiki/User_Datagram_Protocol

[29] TAKAHASHI, Kazuhito. Hand-gesture-recognition-using-mediapipe. In: *Github.com* [online]. 2023 [cit. 2023-05-14]. Dostupné z: https://github.com/Kazuhito00/hand-gesture-recognition-using-mediapipe/tree/main

[30] TensorFlowJs models: Hand Pose Detection. In: *Github.com* [online]. 2023 [cit. 2023-02-14]. Dostupné z: https://github.com/tensorflow/tfjs-models/tree/master/hand-pose-detection

[31] Websocket. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2023 [cit. 2023-04-14]. Dostupné z: https://en.wikipedia.org/wiki/WebSocket

[32] *Tobii* [online]. 2022 [cit. 2023-05-14]. Dostupné z: https://www.tobii.com

[33] Investigation of classroom management skills by using eye-tracking technology - Scientific Figure on ResearchGate. In: *ResearchGate* [online]. [cit. 2023-05-14]. Dostupné z: https://www.researchgate.net/figure/Tobii-Pro-Glasses-2-A-wearable-Eye-Tracker-as-cited-in-tobiiprocom_fig1_346017610

[34] Fitts's law. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2023 [cit. 2023-04-14]. Dostupné z: https://en.wikipedia.org/wiki/Fitts%27s_law

[35] Hansberger, Jeff Peng, Chao Mathis, Shannon Areyur Shanthakumar, Vaidyanath Meacham, Sarah Cao, Lizhou Blakely, Victoria. (2017). Dispelling the Gorilla Arm Syndrome: The Viability of Prolonged Gesture Interactions. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 505-520. DOI:10.1007/978-3-319-57987-0_41.

Příloha A
Protokol testování

Postup pro provedení každého uživatele všemi scénáři tak aby byl přístup pro všechny unifromní a všichni měli stejné podmínky. Tučné písmeno označuje stisk konkrétní klávesy v řídící aplikaci, kurzíva pak instrukce pro operátora.

Uvedení uživatele

- *na plátně černý obraz a na válci prázdné rozhraní*
- *rozsvícený ledpásek ve válci*
- *seznámení s eyetracking brýlemi*
- *nasazení brýlí a jejich synchronizace s kamerou*
- *uživatel se postaví na značku do středu válce*
- *kontrola zda kamera brýlí snímá celé rozhraní*

Seznámení s interakcí s rozhraním

- dejte pokrčenou ruku před sebe dlaní proti kameře
- objeví se ukazatel znázorňující aktuální pozici kurzoru
- zkuste zahýbat rukou a sledovat pohyb ukazatele
- druhou ruku skryjte za sebe, může mást rozpoznávání
- zkuste vyměnit ruce
- v rozhraní se budou nacházet různé objekty, které je možné přesouvat
- *uživateli bude zobrazen testovací obrázek s popiskem* **E**
- uchopení objektu se provede najetím nad objekt a vyslovením příkazu “select”
- uvolnění objektu se provede vyslovením příkazu “release”
- obrázky s popiskem, nacházející se v rozhraní, lze zavírat a otevírat
- zavírat je lze slovem “hide” a otevírat slovem “open”
- následující část se bude skládat z pěti oddělených experimentů
- před každým bude předcházet úvod do daného experimentu a zadání úkolu

Scénář Letiště

- *nastavit černou obrazovku* **R**
- nyní se dostáváte do role cestujícího na letišti který čeká na svůj let do NewYorku
- vaším úkolem bude nalézt svůj let v seznamu letů a jemu přidělené číslo odletové brány
- poté si otevřete mapu a najdete kde se brána nachází
- experiment končí ve chvíli nalezení cesty
- let vám letí až za dlouho, takže nemusíte nikam spěchat a můžete v klidu prozkoumat scénu
- *otevření obrázku letiště*
- *otevření scény letiště v rozhraní* **A**
- *po ukončení experimentu nechat ještě chvilku uživateli pro prozkoumání interakcí*

Scénář Security

- *nastavit černou obrazovku* **R** *a zhasnout ledpásek!!*
- v tomto scénáři budete představovat pracovníka ostrahy, který sedí na vrátnici za sklem a pozoruje kolemjdoucí chodce
- na skle se vám zobrazí portréty postav které hledáte a vaším úkolem bude jich co nejvíce v průběhu najít a říct na jaké ulici se nacházejí
- portréty si můžete pro lepší přehlednost zavírat
- teď si vyzkoušíte testovací kolo s jedním portrétem
- *ukázat portrét* **G**
- *pustit video*
- *nechat dojet video dokud nenajde*
- nyní pustíme hlavní test
- *zobrazit portréty* **H**
- *pustit druhé video*
- *na konci vide ukončit a dát chvilku na odpočinek*

Scénář Shapes 1

- *nastavit černou obrazovku* **R** *a rozsvítit ledpásek !!!*
- v tomto scénáři bude vaším úkolem sestavit co nejrychleji z objektů na skle obrazec, který se vám zobrazí na projekci
- až budete spokojeni řekněte stop
- *zobrazíme objekty* **S**
- *zobrazíme první tvar*
- *zmáčknout* **T** *pro měření času*
- *po vyslovení stop zmáčknout* **T**

Scénář Fitts law task

- *nastavit černou obrazovku* **R**
- v tomto scénáři projdeme několik příkladů a v každém bude vaším úkolem uchopit objekt, co nejrychleji jej přesunout nad cíl a pustit jej
- *zobrazit vzorový příklad* **F**
- objekt je šedý a může mít různou velikost a cíl je vždy červený
- vyzkoušejte si přesun
- nyní na ostro
- *projdeme příklady* **0-9**

Scénář Shapes 2

- *nastavit černou obrazovku* **R**
- *vypnout aplikace s interakcí rukou a hlasem*
- *pustit python interakci gesty*
- nyní si vyzkoušíte opět sestavit obrazec z objektů, tentokrát ale budete moct uchopovat objekty pomocí gest ruky
- otevřená ruka pouze ukazuje
- zavřená ruka uchopí objekt
- *nechat ho si to vyzkoušet na vzorovém tasku* **E**

- *zobrazit tvary* **S**
- *zobrazit druhý obrazec*
- opět až budete spokojeni řekněte stop
- *zmáčknout* **T** *pro měření času*
- *po vyslovení stop zmáčknout* **T**
- *následuje sundání brýlí a uložení logu z experimentu do záznamů s označením uživatele*

Rozhovor

Návodné otázky pro zakončující semistrukturovaný rozhovor:

- Jak se cítíte? Jste vyčerpaný/á?
- Jaký máte pocit z tohoto zážitku?
- Jaké pro vás bylo interagovat s aplikací pomocí rukou?
  - Byly v tom nějaké problémy?
- V souvislosti s prvním experimentem na letišti, dokážete si představit že byste takový průhledný displej a interakce chtěl(a) používat?
- Jaký pocit máte z druhého experimentu?
  - Když byste si podobnou situaci představil(a) probíhající na jedné obrazovce, co vám přijde lepší? A proč?
- Jaké pro vás bylo skládání obrazce?
  - Jaký způsob interakce (gesta, hlas) pro vás byl příjemnější?
- Dokážete si představit smysl v reálném použití podobného průhledného displeje například ve výlohách obchodů, voliérách v zoo atd? Proč?

Příloha B
Zápisy z rozhovorů

První uživatel (muž, cca 30 let, IT/UX)

Po dokončení testování ho lehce bolely ruce a cítil na sobě lehkou únavu z kognitivního vysílení. Zážitek byl pro něj příjemnější než virtuální realita. Dělalo se mu z interakce s displejem méně špatně něž při nošení brýlí. Pozitivně hodnotil odezvu interakce vzhledem k použité technologii. Velký pohyb bez problémů, ale u jemnějšího a přesnějšího pozicování s tím měl problémy. Druhý scénář byl pro něj celkem náročný a trvalo mu než se zorientoval v tom co má dělat.

Poznámky:

- bez předchozího instruování bylo viditelné zmatení uživatele při interakcích na stranách válce, jelikož automaticky předpokládal prostorovost interakcí, tedy i rotoval sám sebe proti stěně válce
- kurzor mu často vylétával za horní okraj obrazovky a musel jej hledat

Druhý uživatel (žena, cca 21 let, studentka medicíny)

Cítila silné vyčerpání rukou z dlouhého natažení při interakci s displejem. Cítila se iritovaná že ji rozhraní neposlouchá (hlasově). Jako příkazové slovo pro uchopení objektu by volila spíše slovo “take”, jelikož je jednodušší a jeho výslovnost se dá obtížněji zkomolit. Všímala si pomalejší odezvy systému nejen na pohyb ruky, ale především na vykonání hlasového příkazu/gesta. Musela tedy dopředu kalkulovat kdy vydat příkaz aby se trefila do cíle.

Dokáže si představit použití průhledného displeje v obchodních centrech jako informačních tabulí a panelů a stejnou formou i na letištích. U druhého scénáře Security si musela trochu poodstoupit či pohnout hlavou aby viděla zobrazené portréty. Dvě vrstvy jí v tomto případě přišli fajn a kdyby si mohla více poodstopit od stěny válce aby měla obě vrstvy ve stejné ohniskové vzdálenosti, pomohlo by jí to v soustředění. Odpadla by nutnost přesouvat a odtrhávat pozornost při přesunu mezi vrstvami. Vzhledem k interakcím, by se jí líbila možnost kombinace obou přístupů, tedy hlasu i gest a možnost je střídat dle vlastní volby.

Poznámky:

- I přes úvodní instrukce o možnosti mít pokrčené ruce, později automaticky přešla do natažených rukou, což vedlo k většímu vysílení.
- Hlasový model ji absolutně neposlouchal. Pouze poté co modulovala hlas na hlubší úroveň podobnou mužskému hlasu. I tak bylo rozpoznávání příkazů stále velmi chybové.

Třetí uživatel (muž, cca 45 let, IT- virtuální a rozšířená realita)

Uživatel byl po skončení experimentu celkem unavený a frustrovaný z prodlevy v odezvě interakcí. Líbilo se mu že nebyla nutná akomodace oka oproti AR brýlím, kvalita projekce mu ale nepomáhá v současném pozorování obou vrstev (rozlišení a kvalita reálné vrstvy i kontrast světlého pozadí a překrývající projekce na průhledném displeji).

Bylo pro něj matoucí sledování ruky, očekával by že kurzor bude sledovat směr vektoru mezi jeho okem a dlaní. Používání rozhraní bylo pro něj únavné a náročné na soustředění. U security tasku by si rád rozestavil objekty jinak než byly aby lépe viděl na obraz. Projekce neměla ideální barvy a kvalitu.

Čtvrtý uživatel (muž, cca 21 let, IT - technik bezpečnostních systémů)

Spokojený se svým výkonem. Necítí žádné vyčerpání. Aplikace i displej ho jako nová technologie velmi zaujaly. Přepínání pozornosti mu nedělalo problém, ale vadilo mu že musel přeostřovat na video se špatným rozlišením, což vyžadovalo jeho větší pozornost. Na základní interakci mu gesta vyhovovaly více než hlasové příkazy. Chápe ale že na komplikovanější příkazy by asi bylo třeba užití hlasu. Kdyby si měl představit užití těchto interakcí na reálných informačních panelech menších rozměrů, přišel by mu dotyk stále přirozenější. Využití takového průhledného displeje by si dokázal představit a dávalo by mu smysl například u velkých komerčních akvárií, kde by se obsah zobrazený na stěnu akvária přizpůsoboval proplouvajícím rybám. Trochu mu vadily brýle, které mu překáželi v plném výhledu. Přesná interakce (přesun objektů) u scénářů s tvary byla velmi náročná.

Poznámky:

- již od začátku zvládal sám od sebe bez problému navigovat kurzor, navíc po celou dobu měl ruku pokrčenou a používal navigaci přímo před sebou místo toho aby dělal velké pohyby s nataženou rukou

Pátý uživatel (žena, cca 24 let, student IT/UX)

Cítila se po experimentu velmi vyčerpaně. Především pak z neustálého držení natažených rukou ve vzduchu a náročnosti na soustředění. Zároveň také díky vydýchanému prostředí.

Vadil jí nepřesný pohyb kurzorem. Měla pozoruhodně obráceně nastavenou intuici. Předpokládala že pokud dá ruku níž, kurzor se posune výše. Stejně tak prohozené měla i strany. Obecně jí interakce přišli neintuitivní a nechtěla by je v normálním životě používat.

Takový průhledný displej by si dokázala představit například v jako informační plochy v obchodních domech a jiných institucích kde se normálně využívají. U Security scénáře měla problém s nutsností přesouvat pozornost mezi oběma vrstvami. Lepší by jí přišlo mít to v jedné vrstvě. Vadila jí nízká přesnost pohybu kurzoru a velká kognitivní zátěž při interakcích spojená s nutností myslet na to, aby nepouštěla gesto pokud chce stále přesouvat objekt. Její intuice fungovala formou “kliknutí”. Taktéž byla překvapena velikostí projekční plochy na válci, na které se jí často ztrácel kurzor.

Poznámky:

- v jejím případě měla aplikace často problém rozpoznat ruku před jejím tělem. Její oblečení ale nebylo nijak specificky odlišné od ostatních pro to aby se mu to dalo přisuzovat.
- zároveň bylo patrné časté zmatení rozpoznávání gest v omylech mezi otevřenou rukou a gestem OK
- stejný problém s rozpoznáváním hlasových příkazů jako u předchozí ženy

Šestý uživatel (muž, cca 24 let, student IT/UX)

Uživatel byl zklamaný ze svého výkonu při jednotlivých scénářích, obecně ale zaujat technologií. Gesta mu vyhovovala více než hlas. Hlasové ovládání pro něj totiž není přirozené. Zároveň vnímal rychlejší odezvu gest než hlasu. Především pak vnímal jako nepříjemný delay u příkazu "release" a náročnost nepohnout rukou při vyslovování takového příkazu. Také si byl vědom fyzické náročnosti na držení ruky ve vzduchu. Líbila by se mu taková myšlenka interakce použitá v informačních panelech jelikož by ji vnímal jako více hygienickou. Nerad sahá na opatlané dotykové obrazovky či navigační tlačítka současných panelů. U Security scénáře mu vadilo překrytí portrétů a videa za nimi. Vnímal nutnost zaměřit se vždy na jednu vrstvu. Preferoval by kdyby navigace kurzoru fungovala tak, že se kurzor objeví přesně tam kam ukáže rukou. Využití reálného displeje si dokáže představit především v hromadné dopravě jako součást oken, kde by mohl vidět informace o aktuální zastávce, trase apod. Zároveň by si zde také dokázal představit zobrazované informace o památkách kolem kterých by projížděl.

Sedmý uživatel (muž, cca 37 let, netechnicky zaměřený UX)

Uživatel po ukončení experimentu cítil znatelné vyčerpání rukou. Frustrovaný z dlouhé odezvy systému na interakce. Taktéž si všímal náročné interakce po okrajích. Z hlediska použitelnosti mu vadila přesnost interakcí a pak také hlasové ovládání a jeho dlouhá odezva, která především u příkazu "release" vyžadovala nehnutí kurzorem, což bylo velmi náročně. Lépe by si takový válcový průhledný displej a především pak interakce dokázal představit v nějakém 3D softwaru kde by se zobrazovaly 3D objekty a ne 2D obsah. Vnímá pozitivum v možnosti mít na jednom místě zobrazeno větší množství informací než na klasickém monitoru. Velký benefit těchto interakcí vnímá také, oproti klasickému dotyku, v hygieně. Srovnává to s upatlanými dotykovými panely v obchodních centrech. Hlas se dle něj použije spíše při nějakém veřejném mluvení či interakci během přednášky ale v soukromí se bude uživatel více spoléhat na gesta. Zároveň zmiňuje obavu o ovlivnění hlasového ovládání okolním ruchem a řečí v pozadí ve veřejných prostorách.

Příloha C

Naměřená data z testování

1 usr

Task	Start X	Start Y	Obj size	Target X	Target Y	Start	End	Duration (m:s)	End X	End Y	End distance
0	1900	100	150	1300	605	11:46:20	11:46:23	**00:03**	1255	695	**100,6**
1	2500	100	250	1950	805						
2	600	800	50	2200	155	11:46:38	11:46:44	**00:06**	2217	177	**27,8**
3	3200	200	250	500	405	11:46:56	11:47:13	**00:17**	541	373	**52,0**
4	3100	700	150	2000	305	11:47:28	11:47:34	**00:06**	1991	263	**43,0**
5	1500	100	150	2300	155	11:47:45					
6	2500	200	250	2000	605	11:47:59	11:48:03	**00:04**	2017	557	**50,9**
7	1500	50	150	2600	905						
8	1850	800	50	2000	155	11:48:13	11:48:19	**00:05**	2035	129	**43,6**
9	2500	100	150	500	805	11:48:28	11:48:46	**00:18**	447	895	**104,4**

2 usr

Task	Start X	Start Y	Obj size	Target X	Target Y	Start	End	Duration (m:s)	End X	End Y	End distance
0	1900	100	150	1300	605						
1	2500	100	250	1950	805						
2	600	800	50	2200	155						
3	3200	200	250	500	405						
4	3100	700	150	2000	305						
5	1500	100	150	2300	155						
6	2500	200	250	2000	605	8:34:53	8:34:56	**00:03**	1889	621	**112,1**
7	1500	50	150	2600	905	8:34:59	8:35:04	**00:04**	2631	941	**47,5**
8	1850	800	50	2000	155	8:35:15	8:35:18	**00:03**	1963	129	**45,2**
9	2500	100	150	500	805	8:35:26	8:35:47	**00:21**	447	783	**57,4**

3 usr

Task	Start X	Start Y	Obj size	Target X	Target Y	Start	End	Duration (m:s)	End X	End Y	End distance
0	1900	100	150	1300	605	10:25:46	10:25:51	**00:05**	1279	575	**36,6**
1	2500	100	250	1950	805	10:25:57	10:26:01	**00:03**	1977	857	**58,6**
2	600	800	50	2200	155	10:26:56	10:27:04	**00:08**	2217	177	**27,8**
3	3200	200	250	500	405						
4	3100	700	150	2000	305	10:27:16	10:27:21	**00:06**	2071	295	**71,7**
5	1500	100	150	2300	155	10:27:29	10:27:35	**00:06**	2231	271	**135,0**
6	2500	200	250	2000	605	10:27:42	10:27:47	**00:05**	2025	629	**34,7**
7	1500	50	150	2600	905	10:27:54	10:27:59	**00:06**	2599	957	**52,0**
8	1850	800	50	2000	155	10:28:09	10:28:14	**00:05**	1987	185	**32,7**
9	2500	100	150	500	805	10:28:23	10:28:30	**00:07**	543	807	**43,0**

4 usr

Task	Start X	Start Y	Obj size	Target X	Target Y	Start	End	Duration (m:s)	End X	End Y	End distance
0	1900	100	150	1300	605	12:31:03	12:31:10	**00:06**	1287	591	**19,1**
1	2500	100	250	1950	805	12:31:18	12:31:23	**00:05**	1961	769	**37,6**
2	600	800	50	2200	155	12:31:45	12:31:51	**00:07**	2217	137	**24,8**
3	3200	200	250	500	405	12:31:58	12:32:07	**00:09**	501	405	**1,0**
4	3100	700	150	2000	305	12:32:19	12:32:25	**00:06**	2023	311	**23,8**
5	1500	100	150	2300	155	12:32:34	12:32:38	**00:04**	2335	167	**37,0**
6	2500	200	250	2000	605	12:32:44	12:32:49	**00:05**	2009	621	**18,4**
7	1500	50	150	2600	905	12:32:54	12:33:03	**00:09**	2583	901	**17,5**
8	1850	800	50	2000	155	12:33:08	12:33:13	**00:04**	1995	129	**26,5**
9	2500	100	150	500	805	12:33:18	12:33:28	**00:10**	471	799	**29,6**

5 usr

Task	Start X	Start Y	Obj size	Target X	Target Y	Start	End	Duration (m:s)	End X	End Y	End distance
0	1900	100	150	1300	605	13:24:18	13:24:21	**00:03**	1159	711	**176,4**
1	2500	100	250	1950	805	13:24:28	13:24:31	**00:03**	1849	889	**131,4**
2	600	800	50	2200	155	13:24:54	13:24:59	**00:05**	2233	209	**63,3**
3	3200	200	250	500	405	13:25:13	13:25:20	**00:07**	509	445	**41,0**
4	3100	700	150	2000	305	13:25:32	13:25:36	**00:04**	2007	495	**190,1**
5	1500	100	150	2300	155	13:25:42	13:25:46	**00:04**	2263	223	**77,4**
6	2500	200	250	2000	605	13:25:56	13:26:01	**00:04**	2073	565	**83,2**
7	1500	50	150	2600	905	13:26:18	13:26:36	**00:17**	2559	1005	**108,1**
8	1850	800	50	2000	155	13:26:58	13:27:03	**00:05**	1883	129	**119,9**
9	2500	100	150	500	805						

6 usr

Task	Start X	Start Y	Obj size	Target X	Target Y	Start	End	Duration (m:s)	End X	End Y	End distance
0	1900	100	150	1300	605	14:19:01	14:19:05	**00:04**	1303	559	**46,1**
1	2500	100	250	1950	805	14:19:11	14:19:16	**00:05**	1897	745	**80,1**
2	600	800	50	2200	155	14:19:29	14:19:40	**00:10**	2217	161	**18,0**
3	3200	200	250	500	405	14:19:46	14:19:58	**00:12**	613	669	**287,2**
4	3100	700	150	2000	305	14:20:10	14:20:16	**00:06**	1879	327	**123,0**
5	1500	100	150	2300	155	14:20:21	14:20:26	**00:04**	2319	159	**19,4**
6	2500	200	250	2000	605	14:20:32	14:20:37	**00:05**	2065	533	**97,0**
7	1500	50	150	2600	905	14:20:45	14:20:51	**00:06**	2575	853	**57,7**
8	1850	800	50	2000	155	14:20:56	14:21:01	**00:05**	2011	161	**12,5**
9	2500	100	150	500	805	14:21:05	14:21:16	**00:11**	519	823	**26,2**

7 usr

Task	Start X	Start Y	Obj size	Target X	Target Y	Start	End	Duration (m:s)	End X	End Y	End distance
0	1900	100	150	1300	605	15:27:25	15:27:28	**00:03**	1295	567	**38,3**
1	2500	100	250	1950	805	15:28:00	15:28:05	**00:06**	1977	889	**88,2**
2	600	800	50	2200	155	15:28:22	15:28:33	**00:11**	2169	129	**40,5**
3	3200	200	250	500	405	15:29:55	15:30:26	**00:31**	549	349	**74,4**
4	3100	700	150	2000	305	15:30:55	15:31:00	**00:05**	2207	247	**215,0**
5	1500	100	150	2300	155	15:31:09	15:31:12	**00:02**	2383	199	**93,9**
6	2500	200	250	2000	605	15:31:23	15:31:28	**00:05**	2025	549	**61,3**
7	1500	50	150	2600	905	15:32:30	15:32:37	**00:06**	2647	845	**76,2**
8	1850	800	50	2000	155	15:32:49	15:32:53	**00:03**	1883	161	**117,2**
9	2500	100	150	500	805	15:33:19	15:33:49	**00:31**	455	799	**45,4**

	Shape 1			Shape 2		
	Start	End	**Duration**	Start	End	**Duration**
User 1						
User 2						
User 3				10:29:57	10:31:36	**01:39**
User 4	12:27:21	12:29:53	**02:32**	12:34:42	12:38:39	**03:57**
User 5	13:20:31	13:22:12	**01:42**	13:28:34	13:30:02	**01:28**
User 6	14:15:35	14:17:32	**01:58**	14:22:40	14:24:15	**01:35**
User 7				15:36:14	15:39:54	**03:41**
		Average:	**02:04**		**Average:**	**02:28**
		Median:	01:58		Median:	01:39