Aplikace na ovládání eye trackingového zařízení s možností komunikace se softwarem Hypothesis

Mgr. Michaela Helísková

Diplomová práce
2020

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav informatiky a umělé inteligence

Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Mgr. Michaela Helísková**
Osobní číslo:	**A17283**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**Kombinovaná**
Téma práce:	**Aplikace na ovládání eye trackingového zařízení s možností komunikace se softwarem Hypothesis**
Téma práce anglicky:	**Eye Tracking Device Control Application with the Ability to Communicate with Hypothesis Software**

Zásady pro vypracování

1. Popište princip fungování eye trackingových zařízení.
2. Představte dosavadní metody ovládání eye trackingových zařízení a jejich propojení s jinými aplikacemi.
3. Vytvořte vlastní aplikaci k ovládání eye trackingového zařízení.
4. Navrhněte koncepci propojení se softwarem Hypothesis.

Rozsah diplomové práce:
Rozsah příloh:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. ŠAŠINKA, Čeněk, Kamil MORONG a Zdeněk STACHOŇ. The Hypothesis Platform: An Online Tool for Experimental Research into Work with Maps and Behavior in Electronic Environments. *ISPRS International Journal of Geo-Information* [online]. 2017, **6**(12) [cit. 2019-11-11]. DOI: 10.3390/ijgi6120407. ISSN 2220-9964. Dostupné z: http://www.mdpi.com/2220-9964/6/12/407
2. SALVUCCI, Dario D. a Joseph H. GOLDBERG. Identifying fixations and saccades in eye-tracking protocols. In: *Proceedings of the symposium on Eye tracking research & applications – ETRA '00* [online]. New York, New York, USA: ACM Press, 2000, s. 71-78 [cit. 2019-11-11]. DOI: 10.1145/355017.355028. ISBN 1581132808. Dostupné z: http://portal.acm.org/citation.cfm?doid=355017.355028
3. POPELKA, Stanislav, Zdeněk STACHOŇ, Čeněk ŠAŠINKA a Jitka DOLEŽALOVÁ. EyeTribe Tracker Data Accuracy Evaluation and Its Interconnection with Hypothesis Software for Cartographic Purposes. *Computational Intelligence and Neuroscience* [online]. 2016, **2016**, 1-14 [cit. 2019-11-11]. DOI: 10.1155/2016/9172506. ISSN 1687-5265. Dostupné z: http://www.hindawi.com/journals/cin/2016/9172506/
4. HOLMQVIST, Kenneth B. I, Marcus NYSTRÖM, Richard ANDERSSON, Richard DEWHURST, Halszka JARODZKA a Joost van de WEIJER. *Eye tracking: a comprehensive guide to methods and measures*. First published. Oxford: Oxford University Press, 2011. ISBN 978-0-19-969708-3.
5. DUCHOWSKI, Andrew T. *Eye tracking methodology: theory and practice*. Third edition. Cham: Springer, 2017. ISBN 978-3-319-57881-1.
6. HORSLEY, Mike, ed., Matt ELIOT, ed., Bruce Allen KNIGHT, ed. a Ronan REILLY, ed. *Current trends in eye tracking research*. Cham: Springer, 2014. ISBN 978-3-319-34369-3.

Vedoucí diplomové práce: **doc. Ing. Libor Pekař, Ph.D.**
Ústav automatizace a řídicí techniky

Konzultant diplomové práce: **Mgr. Čeněk Šašinka, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **28. listopadu 2019**
Termín odevzdání diplomové práce: **15. května 2020**

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Ve Zlíně dne 9. prosince 2019

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 5. 8. 2020

Michaela Helísková, v. r.
podpis diplomanta

ABSTRAKT

Tato diplomová práce se zabývá eye trackingem a softwarovým řešením ovládání eye trackingového zařízení. Teoretická část práce se věnuje představení zrakového aparátu, který s eye trackingem nedílně souvisí, a popisuje metody a historii samotného eye trackingu. Dále jsou popsána dostupná softwarová řešení eye trackingu a vysvětlen princip fungování výzkumné platformy Hypothesis. Praktická část popisuje řešený problém a osvětluje důvody jeho aktuálnosti. Následuje definice požadavků, vysvětlení důvodů výběru daných nástrojů a postup implementace. Dále je popsán zdrojový kód a proces ověření funkčnosti aplikace a splnění zadaných požadavků. Výstupem práce je funkční aplikace na ovládání eye trackingového zařízení SMI RED 250.

Klíčová slova: eye tracking, pohyby očí, WebSockets, Hypothesis, Python2, iView X, SMI RED 250

ABSTRACT

This diploma thesis is about eye tracking and a software solution for controlling an eye tracking device. The theoretical part of the work is devoted to the introduction of the visual apparatus, which is inextricably linked to eye tracking, and describes the methods and history of eye tracking itself. Furthermore, the available software solutions for eye tracking are described and the principle of operation of the Hypothesis research platform is explained. The practical part describes the solved problem and sheds light on the reasons for its topicality. A definition of requirements, an explanation of the reasons for the selection of the tools and the implementation procedure follows. The source code and the process of verifying the functionality of the application and meeting the specified requirements are also described. The output of the work is a functional application for controlling the eye tracking device SMI RED 250.

Keywords: eye tracking, eye movements, WebSockets, Hypothesis, Python2, iView X, SMI RED 250

Na tomto místě bych ráda poděkovala vedoucímu diplomové práce doc. Ing. Liboru Pekařovi, Ph.D. za velmi lidský přístup a cenné připomínky během celého procesu, a konzultantovi Mgr. Čeňku Šašinkovi, Ph.D. za samotný nápad a jako vždy inspirativní přístup k problému. Upřímné poděkování patří i Dr. Saschovi Tammovi ze Svobodné univerzity Berlín za všechen jeho čas, který věnoval mému uvedení do tématu. V neposlední řadě si velký dík zaslouží také mí kolegové, a především Petra Ondřejková, za vydatnou podporu během celého studia.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Eye tracking jako metoda zjišťování místa pohledu člověka se mezi výzkumníky v posledních dekádách těší velké oblibě napříč všemi obory – od marketingu přes sport, psychologii, webový design, kartografii, nebo neurofyziologii. Hlavní důvod rostoucího zájmu plyne ze snahy pochopit, kam lidé soustředí svůj pohled potažmo svoji pozornost. Eye tracking k tomu poskytuje kvantitativní a objektivní informace o průběhu zpracování podnětového materiálu, což dává možnost vhledu do toho, co lidé nacházejí zajímavým a jak vnímají viděné [1].

Popularita eye trackingu je na vzestupu také z dalšího důvodu, a to klesající ceně a z toho plynoucí lepší dostupnosti této výzkumné metody. Zatímco cena komerčních řešení se může vyšplhat až na 10 tisíc dolarů, open source nástroje lze pořídit za mnohem snesitelnější cenu. Cena těch nejlevnějších z nich se v druhé polovině roku 2019 pohybovala v rozmezí 100 až 250 dolarů [2]. Trh však nabízí nejen široký rozsah cen, taktéž technické spektrum samotných zařízení je velmi široké. Existují zařízení monokulární i binokulární, statický nebo eye tracker umístěný na hlavě [3].

Co však výzkumníkům (konkrétně psychologům na Masarykově univerzitě) poněkud ztěžuje život je fakt, že k jejich experimentům jim nestačí samotné eye trackingové zařízení, ale využívají také další nezbytné nástroje, jako je například originální výzkumná platforma Hypothesis [4]. Ta je navržena tak, aby dávala prostor pro tvorbu složitých testových metod, které často v počítačové podobě vůbec neexistují. Počítačová administrace jim však kromě nesporných výhod (přesné měření reakčního času, jednoduché a přehledné ukládaní dat umožňující jejich export, možnost hromadné administrace atd.) přináší i značná úskalí v následném zpracování, spárování a vyhodnocení dat ze dvou nástrojů zároveň. Cílem této práce je proto vytvořit eye trackingovou aplikaci, která by dávala možnost jednoduchého ovládání eye trackingového zařízení a zároveň dávala prostor pro propojení se softwarem Hypothesis v jeden celek.

I. TEORETICKÁ ČÁST

1 EYE TRACKING

Jak napovídá název, eye tracking (zkratka ET) je proces snímání a sběru dat o aktivitě očí, která je reprezentována očními pohyby. Princip fungování lidského vidění je založen na snaze oka zaostřit na podnět neboli přivést vizuální stimul do centra nejostřejšího vidění, k čemuž slouží sakadické pohyby, takzvané sakády [5], které jsou dále vysvětleny v kapitole 1.1.2.2. Jedná se o velmi rychlé pohyby oka, kdy zrak člověka přeskakuje z místa na místo několikrát za sekundu. Pro eye tracking však ve skutečnosti není důležitý samotný pohyb oka jako spíš fáze, kdy oko zůstává v klidové fázi a zpracovává informace o pozorovaném objektu. Tyto fáze se nazývají fixace a mohou trvat od deseti milisekund až po několik vteřin. Přístroj, který měří a zaznamenává údaje o oční aktivitě, se nazývá eye tracker. S jeho pomocí lze určit kam, jak dlouho a kterým směrem jedinec upírá svůj zrak, a to na principu infračerveného záření. To je vysíláno na rohovku a jeho odraz je následně porovnán s polohou středu zornice [3].

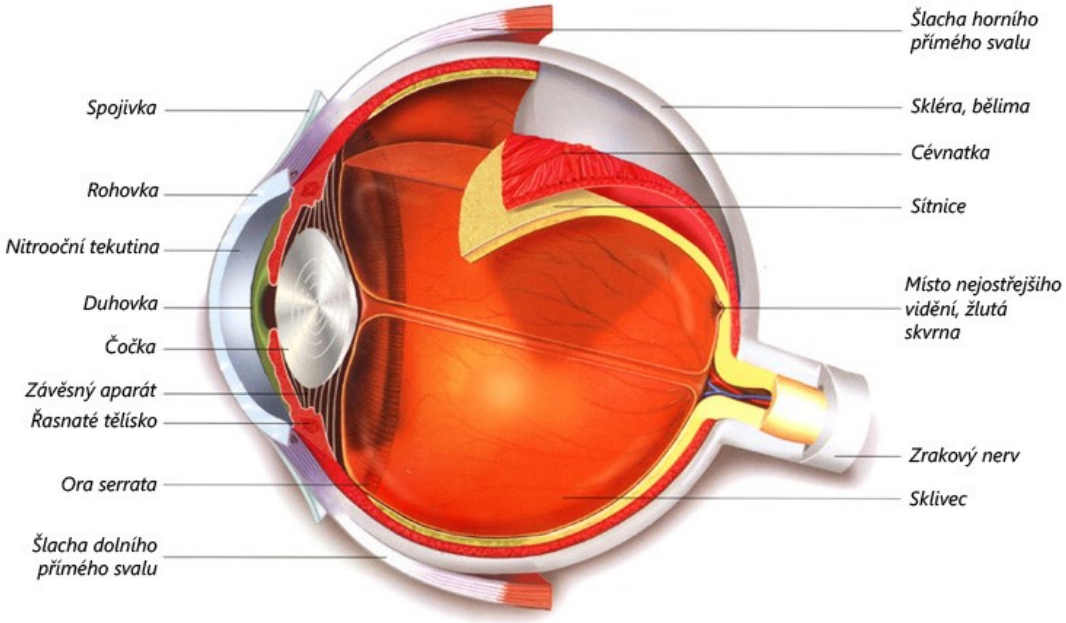
1.1 Zrakový aparát

Pro správné pochopení a následné využití metody eye trackingu je nezbytné mít alespoň základní znalost fungování lidského oka, a to jak z hlediska jeho anatomie jako orgánu, tak i ze strany porozumění okohybným svalům, zajišťujících pět základních pohybů oka.

1.1.1 Anatomie lidského oka

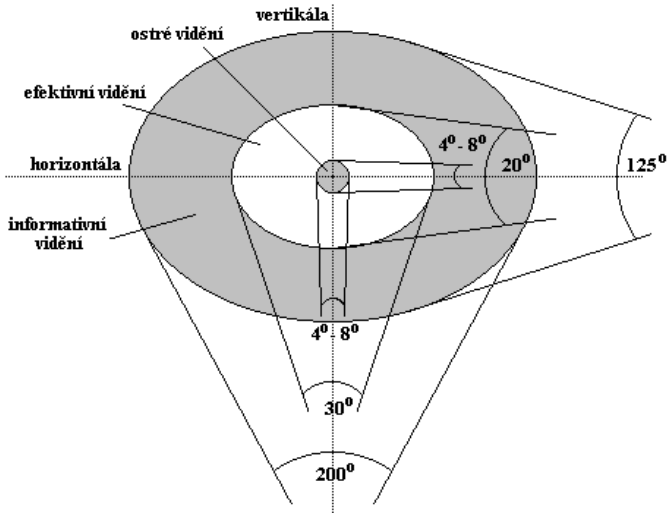
Lidské oko se svojí strukturou plně podřizuje základní potřebě vidění, a to směrování paprsku světla na sítnici (viz Obrázek 1). Zároveň je třeba průchodu co největšího množství světla, z toho důvodu jsou všechny části oka, skrz které světlo prochází, průhledné. Světlo se do lidského oka dostává přes rohovku, za kterou je přední komora oční vyplněná nitrooční tekutinou. Následně dopadá přes zornici na čočku, která je nezbytná pro lom světla a akomodaci – dva faktory ovlivňující, kam přesně paprsek světla nakonec dopadne a jak ostré tedy bude vidění. Po průchodu zornicí se obraz otočí a dopadne na vnitřní a nejdůležitější část oka, což je sítnice. Ta obsahuje velké množství buněk citlivých na světlo, tyčinek a čípků, které převádějí přicházející světlo na elektrický signál, jež je optickým nervem posílán do části mozku zpracovávající vizuální podněty. V jednom místě sítnice se nachází oblast nazvaná žlutá skvrna. Jedná se o místo, kde je extrémní koncentrace čípků v porovnání s periférií sítnice, kde je jejich koncentrace velmi řídká. To má za následek, že je lidské oko

schopno ostrého vidění jen v tomto malém bodě o velikosti nehtu na palci přibližně ve vzdálenosti natažené paže. Pokud tedy chceme na nějaký objekt zaostřit, například na tohle slovo, musíme pohnout očima tak, aby světlo dopadalo právě na žlutou skvrnu [3].



Obrázek 1: Anatomická stavba oka [6]

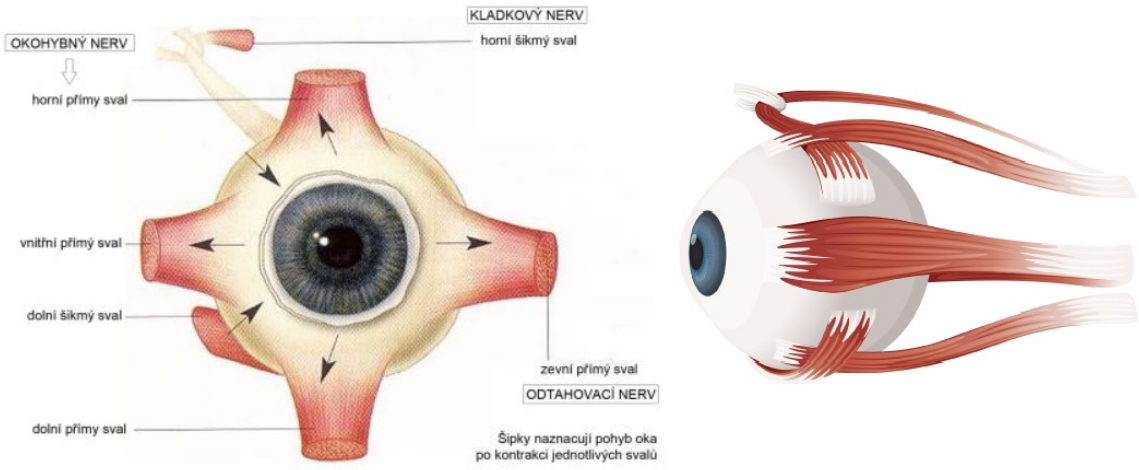
Lidské oči v klidové pozici při fixaci na jeden bod přímo před sebou pokryjí vizuální pole přibližně v rozsahu 95° temporálně, směrem k nosu 65°, vertikálně pak přibližně 130°, a to 60° nahoru a 70° dolů. Jedná se o oblast nazvanou zorné pole, tedy takovou oblast, kterou je lidské oko schopno vnímat vzhledem ke stavbě optického aparátu oka a jeho umístění v lebce. Zorná pole každého z oka se parciálně překrývají a v rozsahu 120° umožňují tzv. binokulární vidění, které je velmi důležité pro vytváření prostorového vjemu. Oblast mimo binokulární vidění, tedy takové, kam má vizuální dosah pouze jedno z očí, se nazývá monokulární vidění [5]. Celkové úhlové pokrytí očí je eliptického tvaru a je v rozsahu 200° na šířku a 130° na výšku [7]. Oblast nacházející se po okrajích tohoto rozsahu (a zároveň po okrajích sítnice) se nazývá periferní vidění. Lidé mají všeobecně špatné periferní vidění postrádající hlubší detaily, barvy i tvary. To je způsobeno právě výše zmíněným řídkým osazením okrajových částí sítnice očními buňkami [8]. Jak lze vidět na Obrázek 2, při přímém pohledu před sebe spadají přibližně 4 stupně z vizuálního pole do oblasti s nejostřejším viděním, tedy na žlutou skvrnu. Ta se latinsky nazývá *fovea* a odtud také pochází název foveální fixace nebo též centrální vidění.



Obrázek 2: Oblast centrálního a periferního vidění [9]

1.1.2 Oční pohyby

Pohyby očí zajišťují tři páry svalů (Obrázek 3), které poskytují lidskému oko stejné možnosti, jako mají oči téměř všech primátů. Jedná se o pět základních pozičních druhů očních pohybů: sakadické (spolu s fixacemi), pomalé sledovací, vestibulární, vergence a fyziologický nystagmus. Kromě toho můžeme rozeznat ještě nepoziční pohyby, jako je akomodace a adaptace čočky, které však nejsou pro potřeby eye trackingu tak důležité [1].



Obrázek 3: Okohybné svaly zajišťující pohyby očí [10]

1.1.2.1 Fixace

Pojem fixace je pro eye tracking nejdůležitějším pojmem vůbec. Jedná se o velmi krátkou dobu (v řádech milisekund až sekund) zafixování obrazu pozorovaného objektu v místě žluté skvrny a funguje reflexně. Jelikož se jedná o fixaci v oblasti zájmu, měření množství fixací

de facto měří množství pozornosti věnované danému stimulu [3]. Aby mohlo k fixačnímu reflexu vůbec dojít, je třeba splnit tři základní podmínky, kterými jsou správná funkce fovei, oblast k fixaci nesmí mít homogenní rysy a fixační podnět se musí shodovat se zájmem pozorovatele [11]. Jakkoliv slovo fixace evokuje úplné zastavení pohybu oka, úplně tomu tak není. Samotná fixace je tvořena zřetelnými mikropohyby: tremor, mikrosakády a drift [12]. Tremor je velmi malý oscilační pohyb s frekvencí okolo 90 Hz podobný třesu, který je zaznamenatelný vysokofrekvenčním eye trackerem. Drift je výrazně pomalejší plynulý pohyb s malou frekvencí okolo 0,5 Hz, který posunuje pohled mimo centrum fixace. Oproti tomu úlohou mikrosakád je velmi rychlými pohyby navracet oko do původní pozice [13]

1.1.2.2 Sakády

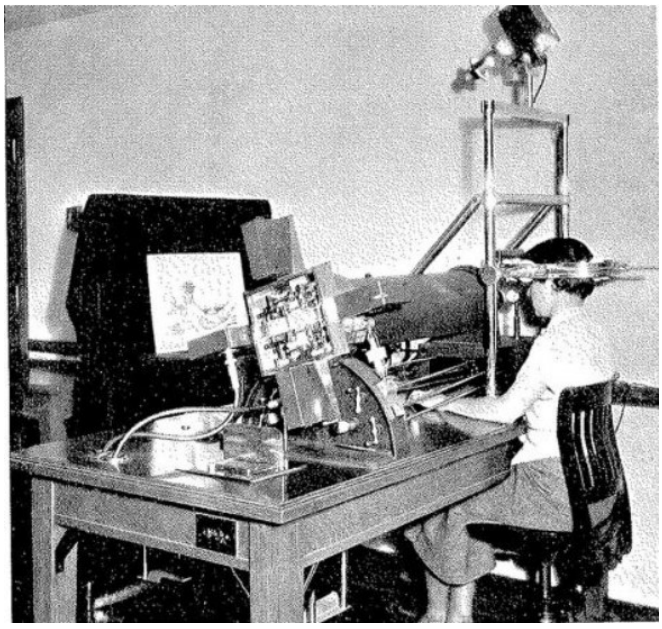
Sakadické pohyby neboli sakády jsou konjugované volní oční pohyby, při kterých dochází k extrémně rychlým (10–100 ms) přesunům očí z jednoho bodu fixace na druhý za účelem prohlížení zorného pole. Časový rozestup mezi jednotlivými sakádami je přibližně 150 ms, což je čas nutný ke zpracování polohy podnětu a vytvoření trajektorie nové dráhy pohybu oka. Jedná se o nejrychlejší pohyb, kterého je lidské tělo vůbec schopné [1]. Během sakadických pohybů je obraz promítaný na sítnici posunutý, což způsobuje rozmazání pohledu. Zároveň jsou tyto pohyby nevědomé, neboť náš vizuální systém aktivně potlačuje vnímání pohybem vyvolaných změn vizuálních stimulů z důvodu udržení percepční stability. Tento mechanismus závisí na přechodném narušení vizuálního zpracování v okamžiku nástupu sakády, takzvaném sakadické potlačení (*saccadic suppresion*), které je podmíněno neurologicky. Díky tomuto mechanismu například nejsme schopni pozorovat pohyb vlastních očí v zrcadle [14].

1.1.2.3 Pomalé sledovací pohyby

Jak napovídá název, pomalé sledovací pohyby jsou určeny k vizuálnímu pronásledování pomalu se pohybujícího podnětu (například letícího letadla na obloze), které jsou naprosto rozdílné od sakád a to včetně částí mozku, které tyto pohyby zabezpečují [3]. Zatímco sakády nastávají i při pohledu na bílou zeď nebo v naprosté tmě při absenci vizuálního podnětu, pomalé sledovací pohyby ke svému vzniku vyžadují pozornostní cíl. V závislosti na rozsahu pohybu cílového objektu jsou pak oči schopny plně přizpůsobit rychlost pohybu rychlosti pohybujícího se objektu [1].

1.2 Metody eye trackingu

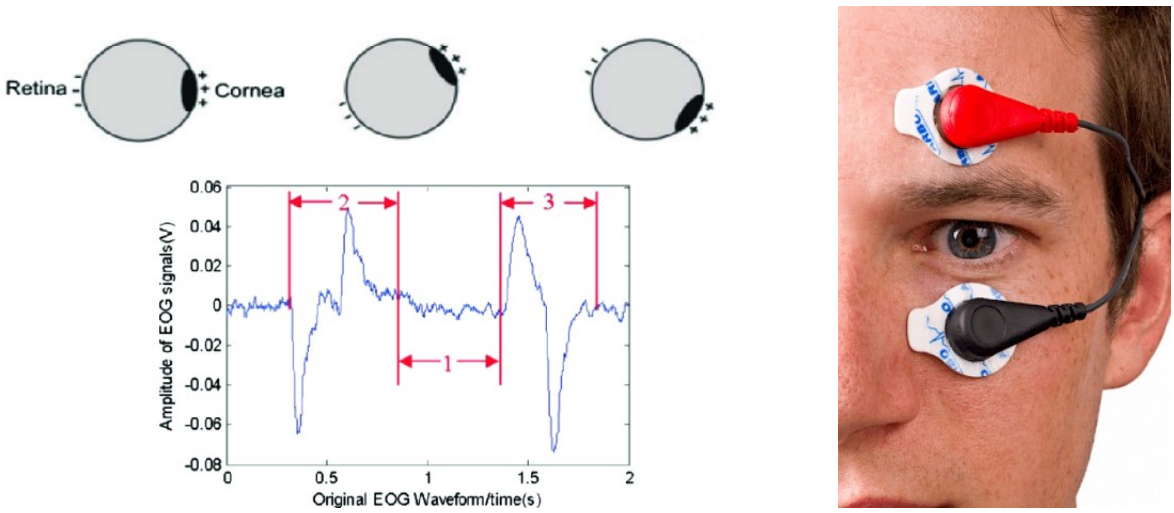
Ačkoliv v dnešní době nejvyužívanější metoda eye trackingu byla představena zhruba před 40 lety, první pokusy zaznamenání toho, kam lidé soustředí svoji vizuální pozornost, lze datovat až na konec 18. století. Stejně tak bylo v té době oftalmologem Luisem É. Javalem učiněno pozorování, které je platné dodnes. Jednalo se o zjištění, že oční pohyby nejsou plynulé, ale lze identifikovat dva rozdílné druhy – fixace a sakády [15]. Eye trackery tehdy však byly kvůli nedostatku pokročilých technologií téměř výhradně mechanické a pro účastníky bylo jejich použití značně nepříjemné, neboť bylo třeba eliminovat nežádoucí pohyby hlavy. Například Huey použil v roce 1898 kousací náustek, který zafixoval pečetícím voskem a tím plně zafixoval hlavu v jedné pozici. Jinou oblíbenou metodou zafixování oka bylo kapání roztoku s kokainem na oční bulvu [3]. Zlom představoval rok 1901 a první neinvazivní metoda, se kterou přišli Dodge a Cline [16]. Napadlo je, že není potřeba připevňovat přístroj na hlavu nebo k oku, ale stačí namířit na oko paprsek světla a následně vyfotografovat jeho odraz. Dali tak vzniknout velmi důležité a stále využívané metodě. Jedno ze zařízení využívané pro tuhle metodu lze vidět na Obrázek 4. Kromě této metody, nazvané foto-okulografie (POG), lze podle Duchowskeho [1] rozlišit ještě tři další: elektro-okulografie, sklerální kontaktní čočky a metodu sledování odrazu světla duhovky a rohovky. Dané metody se liší způsobem sledování pohybu oka, které jsou dva – jeden způsob měří relativní pozici oka k hlavě a druhá metoda bere do úvahy pozici oka v prostoru, díky čemuž je schopna určit bod, na který je pohled zaměřen neboli *point of regard* [17].



Obrázek 4: Zařízení k fotografování pohybů očí z roku 1935 [18]

1.2.1 Elektrookulografie (EOG)

Tato metoda patří mezi techniky, které nejsou schopné určit přesný bod pohledu oka, ale na druhou stranu jsou schopné zaznamenávat oční pohledy i tehdy, pokud je oko zavřené, což je využitelné například při výzkumu spánku. Díky tomu je metoda stále využívaná, a to i přesto, že se řadí spíše ke starším metodám (největší oblibě se těšila v 70. letech 20. století) [7]. Princip metody je založen na zaznamenávání změn elektrického potenciálu na kůži v okolí oka, které jsou zaznamenávány neinvazivními EOG elektrodami s rozsahem napětí 10 μV–3,5 mV a frekvenčním rozsahem v intervalu 0–100 Hz. Elektrický potenciál oka má zápornou hodnotu na sítnici a kladný pól na rohovce, takže při každé změně oční bulvy dojde ke změně elektrického potenciálu (Obrázek 5) [19].

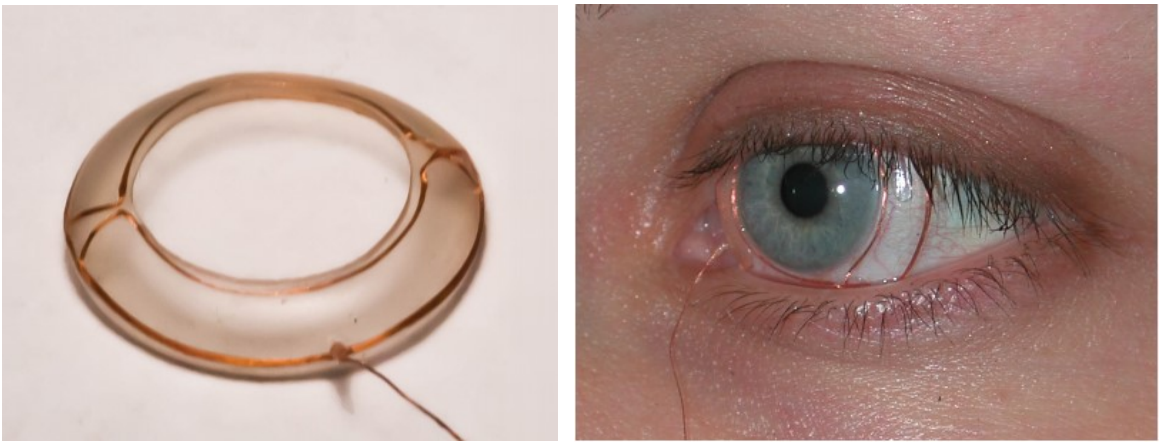


Obrázek 5: Záznam měření metodou elektrookulografie vlevo (nezávislá osa znázorňuje původní tvar EOG signálu v čase, závislá osa pak aplmitudu EOG signálu a to pro pohled rovně, nahoru a dolů) [20] a ukázka vybavení pro EOG, konkrétně se jedná o EOG elektrody ExG Sensor (vpravo) [21]

1.2.2 Sklerální kontaktní čočky

Jak napovídá název, jedná se o metodu, kdy je do oka umístěna speciálně upravená kontaktní čočka obsahující indukční cívku, která zaznamenává změny v elektromagnetickém poli a snímá tak pohyb oka. Díky tomu se metoda řadí mezi jednu z nejpřesnějších eye trackingových metod vůbec a je využívána pro výzkumy v oblasti medicíny [1]. Sklerální čočky jsou schopny zachytit pohyb o malých amplitudách a vysokém temporálním i prostorovém rozlišení. Technika je závislá na jednotném magnetickém poli v oku, které je vytvořeno umístěním hlavy mezi Helmholtzovými cívkami. Čočka se skládá z kroužku vytvořeného z tenkého

kovového drátku, který je zapuštěný do silikonu a umístěn na bělmu oka. Z cívky vede tenký drát, který je připojen k měřícímu zařízení. Při změně orientace oka magnetické pole indukuje napětí ve sklerální cívce. Následným měřením a vyhodnocením velikosti indukovaného napětí systém vyhodnotí pozici oka směrem k hlavě. Jak je patrné z popisu, nejedná se o metodu uživatelsky příliš přívětivou [22].



Obrázek 6: Vlevo sklerální čočka s indukční cívkou před umístěním do oka [22], vpravo stejná čočka po umístění [23]

1.2.3 Fotookulografie (POG) a videookulografie

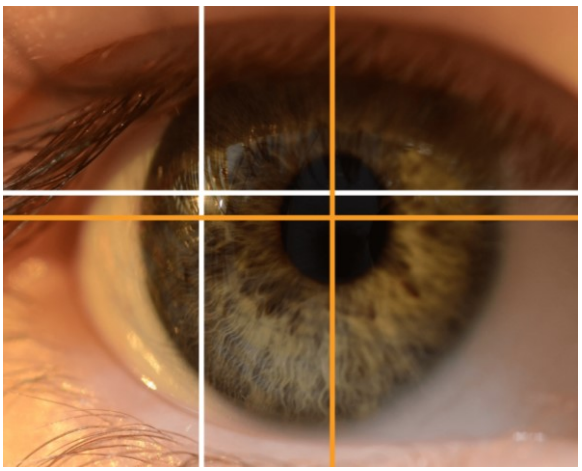
Jedná se o kategorii seskupující více metod dohromady. Jejich podstatou je snímkování nebo natáčení všech očních pohybů, jako je rotace, translace, akomodace nebo sledování odrazů blízko umístěných světelných zdrojů (typicky infračervených) na rohovce [3]. Poslední zmíněné metodě je věnována samostatná další kapitola, neboť na jejím principu funguje valná většina dnešních komerčních eye trackerů.

Vyhodnocování získaných video záznamů lze jen těžko automatizovat a jejich manuální interpretace je nejen velmi časově náročná (je potřeba videozáznam procházet snímek po snímku), ale především nepřesná a náchylná k chybám [1].

1.2.4 Sledování středu zornice a odrazu světla od rohovky (Video-Based Pupil Centre Corneal Reflection)

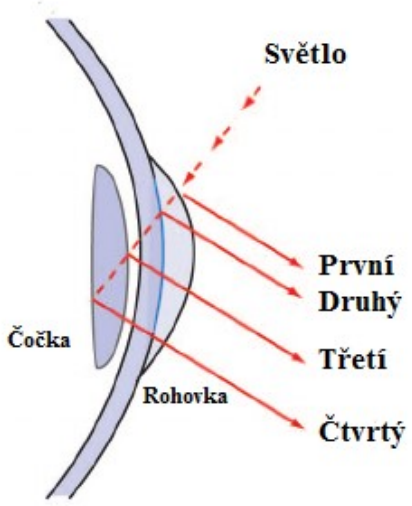
Jedná se o metodu v současné době nejvyužívanější, která zároveň dává možnost kromě pozice oka určit i směr pohledu. Metoda vychází z videookulografie, kdy jsou oči snímány jednou či více kamerami. Dále je její princip založen na algoritmech, které jsou schopny při

blízkém pohledu na oko díky jeho geometrickým a fotometrickým parametrům zaměřit zornici a vypočítat její střed. Další nezbytný parametr je odraz paprsku (typicky) infračerveného světla od rohovky, zaměření a sledování zornice (konkrétně jejího středu) a odrazu paprsku infračerveného světla od rohovky. Z daných hodnot se následně vypočítá jejich relativní vzdálenost, která se sice mění při změně polohy, ale při drobných pohybech hlavy zůstává přibližně stejná [24].



Obrázek 7: Metoda sledování středu zornice (žlutý křížek) a odrazu světla od rohovky (bílý křížek) [25]

Zdrojem infračerveného paprsku je typicky zařízení umístěné na stole, případně počítači nebo na hlavě sledované osoby tak, aby směřovalo přímo do oka. Odraz od rohovky je nejzářivější, nikoliv však jediný a nazývá se korneální odraz (latinsky rohovka = kornea) nebo také první Purkyňův obrázek. Purkyňovy obrázky jsou celkem čtyři, viz Obrázek 8 [3].



Obrázek 8: Čtyři Purkyňovy obrazy způsobené přicházejícím paprskem světla [24]

Druhů eye trackerů založených na výše popsané metodě je několik a každý z nich je jinak vhodný pro potřeby výzkumníka. Nastavení a technická specifikace eye trackeru využívaná v herním průmyslu bude mít zcela jiné parametry než eye tracker na diagnostiku textu. Podle pozice kamer, zdroje světla i typu datového výstupu lze rozlišit následující druhy eye trackerů:

- *statický eye tracker* – jedná se o typ zařízení, u kterého jsou zdroj infračerveného záření i kamery nahrávající pohyb umístěny na stole před zkoumanou osobou, stejně jako monitor s vizuálními podněty. Součástí zařízení může být i konstrukce omezující pohyb hlavy
- *eye tracker umístěný na hlavě* – v tomto případě jsou všechny komponenty umístěny přímo na hlavě participanta, a to například jako helma, čepice nebo eye trackingové brýle. K danému zařízení je možné ještě připojit zařízení na trackování polohy hlavy (head tracker) [3]

2 STÁVAJÍCÍ SOFTWAROVÁ ŘEŠENÍ ET

Pro použití eye trackingového zařízení je kromě samotného hardwaru nezbytné také softwarové řešení. Typicky jde o software zajišťující běh a ovládání zařízení, dále se pak jedná o programy k sestavení samotného experimentu a programy k následné analýze posbíraných dat. Některé z programů nabízejí vícero z vyjmenovaných funkcí dohromady, pro potřeby této práce však bude softwarovým řešením myšlena první popsaná funkcionalita, tedy zajištění samotného běhu eye trackeru. Příkladem budiž eye tracker SMI RED-m 250 od společnosti SensoMotoric Instruments (SMI). K použití eye trackeru slouží program iViewX, který je blíže popsán v kapitole 2.1, k přípravě experimentu software SMI Experiment Center a k vyhodnocení dat SMI BeGaze [7].

Kromě popsaného komerčního řešení je v současné době na trhu k dispozici velké množství open-source aplikací, které bývají častokrát dodávány přímo se zakoupeným eye trackerem. Hlavní rozdíly jsou v použitém programovacím jazyce, dostupnosti zdrojového kódu, podporovaných eye trackerech, platformách a operačních systémech a funkčnosti a spolehlivosti aplikace. Některá ze softwarových řešení jsou představena v následujících kapitolách.

2.1 iView X

iView X je komerční řešení navržené pro tvorbu eye trackingových studií napříč různými obory, od psychologie a neurověd až po využitelnost zdrojů a marketing. Je možné jej využít jak pro rozličné druhy eye trackerů, tak i pro komplexnější aplikace jako fMRI nebo EEG. Systém vytvořila německá firma SensoMotoric Instruments (SMI), která vznikla v roce 1991 jako vedlejší produkt akademického a lékařského výzkumu na Svobodné univerzitě Berlín [26]. V roce 2017 byla firma prodána společnosti Apple a vývoj a podpora velké části eye trackingových řešení byla nejdříve ukončena, nyní se však opět rozbíhá [27].

Jedná se o komplexní systém, který je použitelný jak při práci s jedním PC, tak při duálním nastavení, a je využitelný s širokým spektrem kamerových systémů. Ve všech případech se však jedná o originální zařízení společnosti SMI. Ke spuštění iView X softwaru je třeba mít zakoupenou licenci, která je vydávána vždy jen pro jeden počítač se specifikovaným sadou příslušenství [26]. Více je celá aplikace přiblížena v praktické části.

2.2 GazeParser / SimpleGazeTracker

GazeParser je open-source multiplatformní knihovna využitelná pro sledování pohybů očí a jejich následnou analýzu a vizualizaci. Jedná se o projekt japonského studenta z roku 2012, který se však dále vyvíjí a na stránkách projektu přibývají nové aktualizace. Nejnovější verze 0.11.1 byla vydána v prosinci 2019.

Knihovna se skládá ze dvou komponent, a to SimpleGazeTracker a GazeParser. První zmíněná komponenta je zodpovědná za zachycování obrazu z kamery snímající pozici oka a je psaná v jazyce C++. Druhou komponentou je knihovna psaná v jazyce Python, která zabezpečuje kalibraci, synchronizaci prezentovaných stimulů a nahrávání a analýzu očních pohybů. GazeParser využívá balíky OpenCV, SciPy a Matplotlib a byl primárně vyvinut především pro operační systém Windows, avšak postupem času byla přidána i podpora systémů Linux a MacOS X. Podporovaným hardwarem jsou kamery OptiTrack V 120 a V100R2, CameraLink, Point Grey Flea3 a případně jakékoliv další kamery, které podporuje knihovna OpenCV [28].

Pro správné hardwarové nastavení jsou nezbytné dva počítače, software bohužel nemůže být použit jen s jedním zařízením. Prvním počítačem je tzv. Record PC, který zajišťuje získávání očních pohybů a vypočítává místo pohledu. Na stejném počítači bude nainstalován SimpleGazeTracker. Druhý počítač (Presentation PC) slouží k prezentaci podnětového materiálu účastníkům výzkumu.

Využití knihovny GazeParser by se dalo nazvat poměrně nízko-nákladovým, a to obzvlášť, pokud ji porovnáváme s komerčním řešením iView X. Vstupními náklady jsou pořízení zdroje infračerveného světla, kamery na snímání pohybu či strojového vidění a opěrky brady a hlavy, která zamezí nežádoucímu pohybu hlavy. Na druhou stranu však dává využití GazeParseru uživateli značnou flexibilitu, protože může využít kamerový systém dle vlastního uvážení a jeho výběr není omezen nabídkou podporovaných zařízení [29].

2.3 PyGaze

PyGaze je open-source softwarový balíček, určen pro tvorbu designově komplexních eye tracking experimentů. Na vytvoření PyGaze se v roce 2014 podíleli tři výzkumníci z univerzit Oxford, Aix-Marseille a Utrecht. PyGaze nabízí možnost prezentace vizuálních a sluchových podnětů, online detekci pohybu očí s použitím upraveného algoritmu nebo záznam odezvy skrze klávesnici, myši, joystiku nebo jiného externího hardwaru.

Celá aplikace je napsaná v programovacího jazyce Python s důrazem na to, aby byla uživatelsky co nejpřívětivější a uživatel tak musel vynaložit co nejmenší úsilí při použití nástroje a čtení skriptu. Zároveň je zachována vysoká funkcionalita a flexibilita aplikace, která je využitelná na všech operačních systémech [30].

Pro eye tracking experiment za použití PyGaze je potřeba instalace alespoň dvou externích balíčků, přičemž jeden slouží pro komunikaci s eye trackerem a druhý po experimentální procesy v podobě kompletních knihoven PsychoPy nebo PyGame. Aplikace má velmi široký záběr kompatibilního hardwaru a lze ji využít eye trackery značky EyeLink, SMI nebo Tobii systems bez nutnosti jakékoliv úpravy kódu [31].

2.4 ITU Gaze Tracker

ITU Gaze Tracker představuje další z řady nízko nákladových open-source systémů vytvořených pod záštitou univerzity, v tomto případě ITU univerzity v Kodani. První verze aplikace byla vydána v roce 2009, nyní je však již další vývoj ukončen, protože autoři projektu se začali věnovat projektu novému, a to Eye Tribe [32].

Zdrojový kód je psaný v jazyce C# a využívá knihovnu OpenCV pro zpracování obrazu. Dalšími využitými knihovnami jsou Emgu a AForge. Aplikace se skládá ze tří hlavních komponent a to (1) knihovny pro sledování pohledů očí, která implementuje všechny způsoby ovládání sledovacího zařízení, jako je extrakce očních prvků, spuštění kalibrace, odhad souřadnic pohledu a detekce typu pohybu očí (fixace a sakády); (2) třídy kamery, která je odpovědná za inicializaci generické kamery a pořízení záběrů, které jsou poté zpracovány knihovnou pro sledování pohledů; a (3) uživatelského rozhraní, které poskytuje komunikaci s knihovnou pro sledování pohledů za účelem nastavení různých parametrů systému. Požadovaným operačním systémem je Microsoft Windows XP a vyšší s nainstalovaným frameworkem C#.NET [32].

ITU Gaze Tracker podporuje jak umístění kamery na stole, tak upevnění kamery na hlavě a je možné zvolit jak monokulární, tak binokulární režim.

3 HYPOTHESIS

Software Hypothesis je originální výzkumná platforma vyvinuta primárně pro potřeby testování účastníků výzkumů v kartografii a pro psychologickou diagnostiku, která vychází ze starší verze nazvané MuTeP (multivariantní testovací program) [33]. Software vznikal na Masarykově univerzitě v rámci Centra experimentální psychologie a kognitivních věd (CEPSoS) a dále se na jeho vývoji podílely Přírodovědecká a Filozofická fakulta a firma Tilioteo. Platforma umožňuje tvorbu a adaptaci rozličných diagnostických metod, které jsou jinak dostupné v podobě papír-tužka. Počítačová administrace poskytuje extrémně přesné měření reakčního času (v řádech milisekund), jednoduché a přehledné ukládání dat umožňující jejich export a volnou manipulaci a v neposlední řadě dává také možnost hromadné administrace. Díky tomu, že se jedná o webovou aplikaci, je možné ji využít vzdáleně kdekoliv ve světě s dostatečně rychlým internetovým připojením.

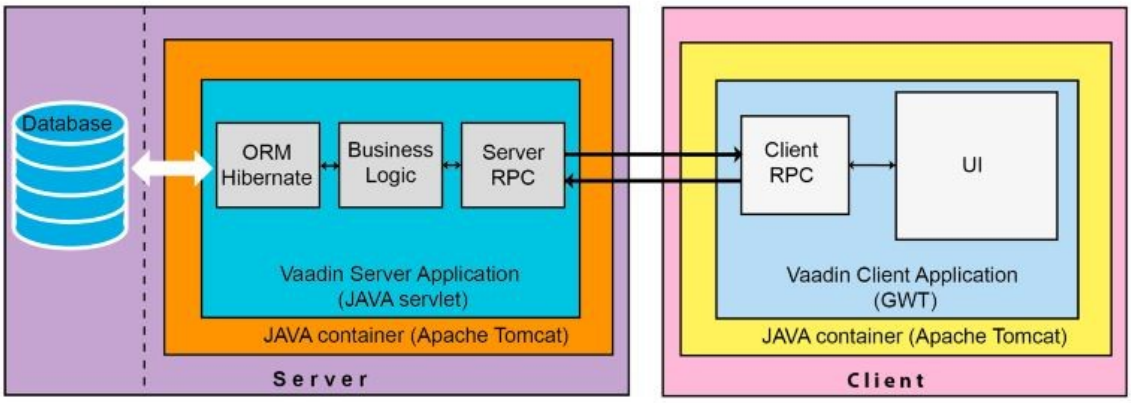
Kromě experimentálního využití je i komerčně nasazen, a to konkrétně ve Vojenské nemocnici v Brně a Ústřední vojenské nemocnici v Praze, kde se využívá pro testování uchazečů o službu v armádě. Druhá zmíněná nemocnice zakoupila licenci na čtyřleté využívání a trojici testů kognitivních funkcí [34].

V následujících kapitolách bude často použit termín *testování,* avšak nebude se jednat o pojem ve smyslu známém z vývoje softwaru, tedy proces kontroly kódu a zajištění kvality aplikace s absencí chyb, ale bude zde mít význam psychologického testování čili diagnostiky za pomoci diagnostických metod.

3.1 Technický design a architektura

Uživatelské rozhraní včetně výkonného jádra je postaveno na frameworku Vaadin 7, použitý databázový systém je objektově-relační systém PostgreSQL (verze 9.1.) a práci s ním zajišťuje ORM Hibernate. Architektura aplikace má tři vrstvy a to klient, server a databáze. Klientská část je určena ke komunikaci a interakci s uživatelem a její běh je zabezpečen klasickým webovým prohlížečem anebo speciálním prohlížečem Hypothesis Browser, který je součástí aplikačního balíčku. Daný prohlížeč dává možnost striktnějšího zabezpečení podmínek pro provádění testů a je vystavěný na komponentách Standard Widget Toolkit. Komunikace mezi klientskou částí a serverem zabezpečuje Ajax RPC běžící na pozadí. Serverová část je aplikována v podobě servletu aplikačního serveru Apache Tomcat zpracovávající požadavky klienta a podle nich následně updatuje uživatelské rozhraní. Použitá knihovna

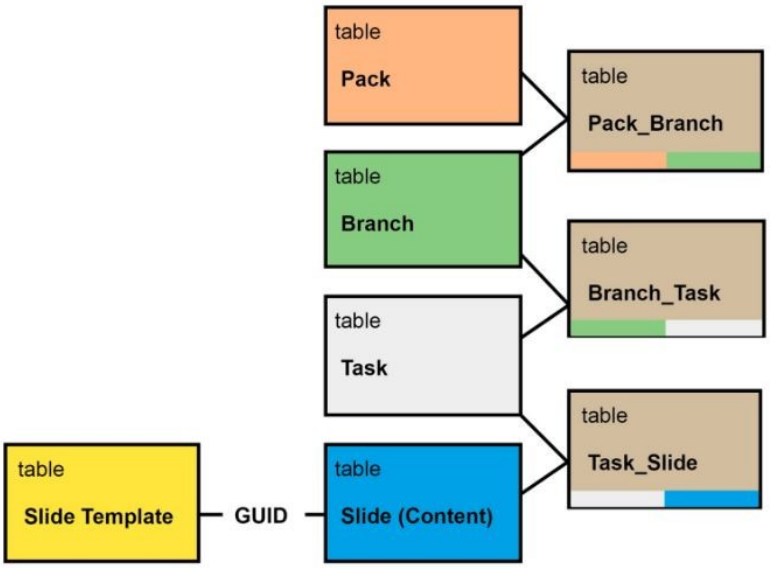
Hibernate zajišťuje s využitím metod objektového mapování entit komunikaci mezi servletem a databázovým systémem. Hibernate podporuje připojení ke všem běžně využívaným databázovým systémům na základě jednotného rozhraní [35]. Schéma technické struktury celé aplikace je naznačeno na Obrázek 9.



Obrázek 9: Schéma technické struktury Hypothesis [35]

3.2 Databázová struktura

Jednotlivé testové baterie jsou v databázi uloženy strukturovaně. Na nejvyšší úrovni hierarchie je tabulka *pack* neboli v podstatě test samotný, pod ní je pak tabulka větve *branch*. Následuje jedna nebo více tabulek úkolů *task*, kdy každá obsahuje alespoň jednu tabulku scény *slide*.



Obrázek 10: Struktura uložení tabulek v databázi [35]

Tabulky *slide* jsou obvykle uspořádány lineárně, díky čemuž jsou účastníkovi experimentu prezentovány v pevném pořadí. K dispozici je však i náhodné, stromové nebo cyklické uspořádání. Při spuštění testu je zvolená tabulka načtena z databáze do serverové aplikace a je vytvořena nová instance testu. Tabulky *branch* jsou při běhu testu procházeny lineárně podle interakce ze strany testované osoby, kdy každá interakce je zaznamenávána do logu událostí. Na nejnižší úrovni se nachází již zmíněná tabulka *slide*, skládající se z komponent *slide_template* a *slide_content*, neboli stylové a obsahové šablony. K jedné obsahové šabloně náleží právě jedna stylová šablona. K jedné stylové šabloně však může existovat jedna či více obsahových šablon. Obě jsou popsány ve formátu XML. Po spuštění testu jsou oba prvky složeny do jednoho XML souboru a podle něj jsou vytvořeny prvky uživatelského rozhraní. Každá *slide_template* obsahuje detaily o struktuře a funkční logice dané scény a informace o vizuálním uspořádání komponent a dialogových oken [36].

Průběh testu je ukládán do hierarchické struktury v tabulkách *test* a *event*, které propojuje tabulka *test_event*. Mezi zaznamenávané akce patří jak události ze strany uživatele (stlačení tlačítka), tak i samotným systémem (načtení další stránky). Další tabulkou, která obsahuje výsledky uživatele při jeho průchodu testem je tabulka *slide_output*. Do té jsou zaznamenávány například hodnoty vepsané do dotazníkových polí. Tabulky *role*, *user*, *user_permission*, *group* a *group_permission* slouží ke správě uživatelů, jejich práv a autorizaci [37].

3.3 Princip fungování

Současná verze softwaru má následující funkcionality: časovač (po vypršení časového limitu aplikace ukončí danou scénu a přejde na další úkol v testové baterii), dialogová okna, pole pro text, selekci tlačítka (účastník výzkumu je vyzván k výběru jednoho či více odpovědí), selekci řádku a výběr pořadí, kresba linky, single a multi click (označení právě jednoho či několika elementů na scéně) a jiné.

3.3.1 Počítačové adaptivní testování (CAT)

Platforma Hypothesis umožňuje provádět počítačové adaptivní testování (Computerized Adaptive Testing – CAT) na několika úrovních hierarchie: na úrovni scény (slide), úkolu (task) a větví. Na úrovni scény jsou události events (např. kliknutí myší) párovány s konkrétními akcemi, přičemž proměnným jsou přiřazovány specifické hodnoty podléhající aritmetickým a logickým operacím. Dostupné algoritmy zahrnují větvící algoritmy (IF-THEN-ELSE, SWITCH) a smyčky (WHILE), které jsou využívány k řízení akcí souvisejících se

scénou. CAT optimalizace na úrovni scén obsahuje i za tímto účelem speciálně vytvořenou scénu. Na úrovni úkolů může být kterákoliv scéna ze sekvence po sobě následujících scén vybrána na základě výstupu vztahujícího se k předchozímu snímku a na základě předdefinovaných pravidel. Pravidla určující průběh každého úkolu jsou součástí tabulky Task v databázi. Adaptivní řazení na úrovni větví má podobu volby větve na konci větve předcházející.

3.3.2 Synchronní a asynchronní testování

Synchronní testování vyžaduje master a slave pack. Multi-task režim byl vyvinut pro experimenty, které vyžadují, aby účastník dokončil několik úkolů současně, nebo aby byl schopný pracovat na více obrazovkách zároveň. Multi-task režim byl testován na adaptaci Testu periferního vnímání (PP-R Periphere Wahrnehmung-R) vytvořený Schuhfriedem. Test je součástí Vídeňského testovacího systému (něm. das Wiener Testsystem) a vyžaduje specializovaný hardware. V pilotním experimentu, jehož účelem bylo ověřit funkčnost synchronního testování, byli účastníci vyzváni, aby současně dokončili primární úkol (kliknutím na cílové objekty na obrazovce) a sekundární úkoly na periferních obrazovkách pomocí klávesy klávesnice.

Asynchronní testování neboli multi-player režim byl původně určen pro výzkum v oblasti geografie. Režim umožňuje několika uživatelům současně pracovat na řešení stejného úkolu, kdy každý z testů je ovládán zvlášť daným uživatelem, avšak pokud událost vyžaduje provedení určité operace, mohou nastat pauzy v asynchronitě. Administrátor může definovat podmínky určující průběh testování pro jednotlivé účastníky, například v jakém pořadí se scény zobrazují participantovi, který z účastníků bude testován jako první a podobně. Režim asynchronního testování byl například použit v řadě adaptací experimentů se sociální psychologie nebo behaviorální ekonomiky [35].

3.3.3 Modul Manager

Modul Manager představuje rozhraní, přes které je možné přistupovat k softwaru z několika uživatelských úrovní, které se liší právy. Na nejnižší přístupové úrovni se nachází User (uživatel), který může pouze vybírat a spouštět testy, k nimž mu dal přístup uživatel z vyšší úrovně. Na další úrovni je pak Manager (manažer), který má administrační práva ke své skupině, v rámci které může vytvářet uživatelské účty, spravovat testovací balíčky a přistupovat k výsledkům testování členů skupiny. Zároveň má přístup k modulu exportu, editoru snímků a účtu správce, který umožňuje hromadné vytváření a správu uživatelských účtů.

Nejvýše umístěný v hierarchii uživatelů je Superuser. Ten má neomezená administrační práva ve vztahu ke všem uživatelům, testům i výsledkům. Uživatelé bez účtu mohou využít přístupu jako Guest (host), avšak pouze s omezeným přístupem k bezplatným balíčkům [35].

3.4 Konfigurační XML soubory

Při běhu testovacího prostředí aplikace Hypothesis jsou testovací otázky sestavovány ze dvou XML souborů, z nichž jeden tvoří stylovou a druhý obsahovou šablonu. K jedné obsahové šabloně náleží právě jedna stylová šablona, avšak k jedné stylové šabloně však může existovat jedna nebo víc obsahových šablon. XML soubory je doporučeno vytvářet v kódování UTF-8 [37].

3.4.1 Šablona scény (SlideTemplate)

SlideTemplate je kořenový element, který musí obsahovat každá stylová šablona. Ta musí dále obsahovat povinný unikátní atribut UID, jenž představuje jedinečný identifikátor. Bez těchto informací by aplikace Hypothesis nebyla schopna sestavit testovací scénu. Pro zajištění jedinečnosti je doporučováno vygenerovat řetězec UID pomocí generátoru jedinečných identifikátorů (například http://createguid.com) [37]. Identifikátor daného snímku bude v XML kódu napsán následujícím způsobem:

```
<?xml version="1.0" encoding="UTF-8" ?>
<SlideTemplate UID="CA442B90-6C6B-11DE-8769-03E855D89593">
</SlideTemplate>
```

Dále musí šablona scény obsahovat element *Viewport*, který představuje testovací obrazovku. Její vzhled definují elementy jednotlivých komponent, ze kterých se scéna skládá, např. *Layers*, *HorizontalLayout*, *VerticalLayout*, atd. Další elementy šablony jsou nepovinné:

- *Window* – definuje podobu vyskakovacích oken
- *Timers* – definuje časovače
- *Variables* – definuje proměnné
- *Actions* – definuje akce
- *Handlers* – umožňují nastavit obsluhu akcí

Příklad použití popsaných elementů [36]:

```
<SlideTemplate UID="CA442B90-6C6B-11DE-8769-03E855D89593">
  <Viewport>
    <Panel Id="p1">
```

```
      ...
    </Panel>
    ...
 </Viewport>
 <Windows>
   <Window Id="w1">
     ...
   </Window>
   ...
 </Windows>
 <Timers>
   <Timer Id="t1">
     ...
   </Timer>
   ...
 </Timers>
 <Variables>
   <Variable Id="v1">
     ...
   </Variable>
   ...
 </Variables>
 <Actions>
   <Action Id="a1">
     ...
   </Action>
   ...
 </Actions>
 <Handlers>
   <Init>
     ...
   </Init>
 ...
 </Handlers>
 <OutputValue>
   ...
 <OutputValue>
</SlideTemplate>
```

3.4.2 Obsah scény (SlideContent)

Druhým XML dokumentem je obsah scény pro sestavení testové otázky. Kořenový element obsahu se musí analogicky nazývat SlideContent s povinným atributem TemplateUID. Hodnota atributu se musí shodovat s atributem UID v šabloně, protože jejich hodnoty jsou při sestavování scény porovnávány. Pokud se neshodují, k vytvoření scény nedojde. Dále musí kořenový element obsahovat element Bindings, který obsahuje 0..n elementů Bind, jež obsahuje element povoleného typu, z nichž se skládá scéna [36].

```
<SlideContent TemplateUID="CA442B90-6C6B-11DE-8769-03E855D89593">
 <Bindings>
   <Bind>
     <ButtonPanel Id="selection">
       ...
     </ButtonPanel>
```

```
   </Bind>
   <Bind>
     <Panel Id="p1">
       ...
     </Panel>
   </Bind>
 </Bindings>
</SlideContent>
```

3.4.3 Sestavování scény – kombinování šablony a obsahu

Pro vytvoření scény je třeba propojit element komponenty šablony s elementem obsahu scény, a to pomocí shodného ID tak, aby vznikl výsledný element. Je tedy nutné vždy definovat alespoň prázdný prvek komponenty s daným parametrem ID v šabloně. Jestliže element dané komponenty v šabloně obsahuje některý z elementů Properties, Actions nebo Layers a zároveň příslušný element komponenty v obsahu scény, pak jsou jednotlivé subelementy Property, Action, Layer porovnávány se stejnými sub-elementy v XML obsahu. Pokud se indikátory shodují, je hodnota ze šablony přepsána hodnotou z obsahu. Obsah scény je tedy nadřazen šabloně. Zbylé elementy jsou do výsledného dokumentu pouze zkopírovány [36].

Šablona:

```
<ButtonPanel Id="selection">
 <Properties>
   <Height Value="90%" />
   <Width Value="90%" />
     ...
 </Properties>
 ...
</ButtonPanel>
```

Obsah:

```
<Bind>
 <ButtonPanel Id="selection">
   <Properties>
     <Width Value="80%" />
     <Captions Value="'3','5','7','žádné„ />
     ...
   </Properties>
   ...
 </ButtonPanel>
</Bind>
```

Výsledek:

```
<ButtonPanel Id="selection">
 <Properties>
```

```
    <Height Value="90%" />
    <Width Value="80%" />
    <Captions Value="'3','5','7','žádné„ />
    ...
  </Properties>
  ...
</ButtonPanel>
```

II. PRAKTICKÁ ČÁST

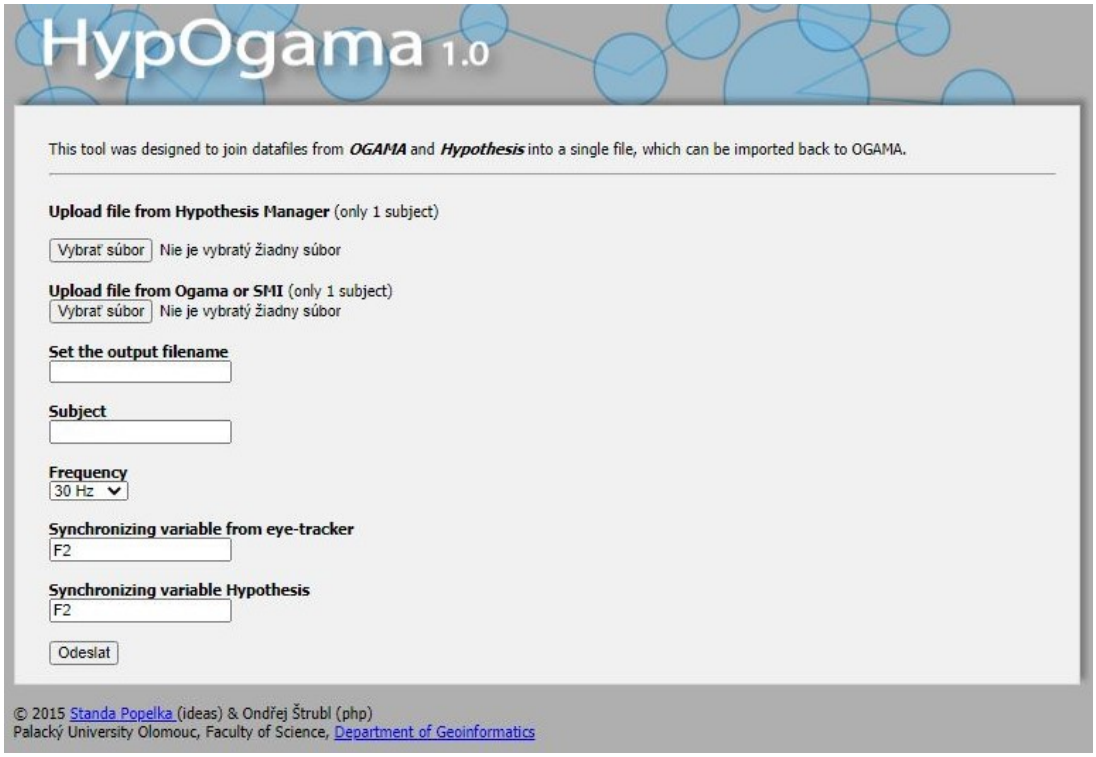
4 ŘEŠENÝ PROBLÉM

Řešený problém vychází z konkrétního požadavku výzkumníků napříč vícero obory na Masarykově univerzitě pod vedením Mgr. Čeňka Šašinky, Ph.D.

4.1 Současná situace

Aplikace na ovládání eye trackingového zařízení, jako je například OGAMA [38], obsahují základní funkcionalitu pro tvorbu podnětového materiálu. Ta však uživateli nedává příliš prostoru k úpravám (není například podporován interaktivní podnětový materiál, není zobrazováno kliknutí myší atd.). Oproti tomu software Hypothesis dává možnost tvorby složitého podnětového materiálu, na druhé straně však ve své aktuální podobě nepodporuje ovládání eye trackingového zařízení. Psychologické a kartografické výzkumy probíhající na Masarykově univerzitě ovšem velmi často vyžadují současného využití obou nástrojů. Řešení, které by efektivně propojovalo eye tracking s Hypothesis, v současné době na trhu neexistuje. Jak program OGAMA, tak Hypothesis po dokončení experimentu vygenerují separátní datový soubor, který bylo dříve třeba manuálně zpracovat a data spárovat. Nahraná data byla rozdělena podle toho, ke kterému slidu v Hypothesis patřila a k nim byla přidána časová známka indikující změnu slidu. Daný postup byl však časově velmi neefektivní s vysokou chybovostí, proto byla vytvořena volně dostupná webová aplikace HypOgama, která celý proces úpravy dat automatizuje. Aplikace má jednoduché uživatelské rozhraní (Obrázek 11), přes které uživatel nahraje exportovaný soubor z Hypothesis a zároveň z OGAMY a zvolí, která klávesa bude synchronizačním klíčem (stlačení konkrétní klávesy slouží k započatí experimentu v Hypothesis a je zaznamenáno oběma programy, Hypothesisem, i OGAMOU) a další potřebné parametry. V dalším kroku aplikace prohledá Hypothesis soubor a vyhledá časové známky přechodu mezi slidy. Tyto časové známky jsou použity k rozdělení neupravených eye trackingových dat do bloků podle toho, ke kterému slidu náleží. Ke každému záznamu každého bloku je přidán název příslušného podnětu. V závěrečném kroku je struktura dat upravena tak, aby nově vytvořený soubor byl přímo importovatelný do nového OGAMA projektu [4].

Přesto, že aplikace HypOgama práci výzkumníků zjednodušuje, stále není plnohodnotným řeším. To představuje kompletní propojení softwaru Hypothesis s eye trackingovým zařízením tak, aby výsledný soubor po ukončení experimentu obsahoval již synchronizovaná data jak z eye trackeru, tak z Hypothesis.



Obrázek 11: Uživatelské rozhraní webové aplikace HypOgama

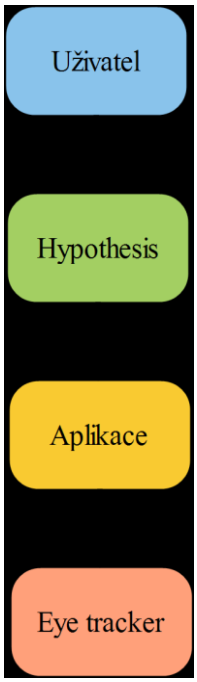
4.2 Požadavky na aplikaci

Definování požadavků je prvním krokem při vývoji softwaru. S ohledem na kvalitu výsledného produktu jde o jeden z nejdůležitějších kroků vůbec, obzvláště u metodiky vývoje softwaru, která je založená na modelu vodopádu. U takového modelu je totiž případná chyba plynoucí ze špatně specifikovaných požadavků objevena až v posledních fázích vývoje, což je testování. Případná chyba v definici tak následně prodlužuje celý proces a v případě komerčního projektu i zvyšuje náklady. Z tohoto důvodu byl zvolen agilní přístup a nové poznatky či překážky byly komunikovány se zadavatelem během celého procesu.

Na samotném začátku byly požadavky specifikovány jen velmi všeobecně: požadovaným výstupem je, aby vznikla aplikace, kterou bude možné propojit se softwarem Hypothesis tak, aby se skrz něj dalo ovládat eye trackingové zařízení. Na základě toho byl navržen diagram použití (Obrázek 12) a vypracován modelový případ. Ten je následovný:

1. Podnětový materiál různého druhu pro psychologický experiment je vytvořen v Hypothesis
2. Účastník experimentu má v Hypothesis vytvořen účet a je přihlášen

3. Po započetí experimentu je uživatel sám schopen kontrolovat eye tracker podle instrukcí na obrazovce
4. Po dokončení experimentu je výstupem soubor, který obsahuje spárovaná data jak z experimentu, tak ze samotného eye trackeru



Obrázek 12: Diagram návrhu použití aplikace

Během procesu vývoje z důvodů popsaných v kapitole 5.4 nebylo možné provést implementaci tak, aby bylo zajištěno propojení s Hypothesis. Ten byl proto z implementace vynechán a požadavky byly přeformulovány tak, aby odpovídaly nově zjištěnému stavu (nově vytvořený diagram použití je znázorněn na Obrázek 13).

Obrázek 13: Upravený diagram návrhu použití aplikace

Z toho vyplynulo následující:

- uživatel bude schopen přes aplikaci ovládat eye trackingové zařízení
- pod pojmem ovládat jsou schovány tyto funkcionality:
    - připojení eye tracker k serveru
    - spuštění kalibrace
    - spuštění validace
    - spuštění nahrávání pohybů očí
    - ukončení nahrávání
    - odpojení eye trackeru
- následně aplikace uloží nahraná data jako textový soubor podle definovaných parametrů (časová známka a souřadnice pohledu levého a pravého oka)
- aplikace bude programovaná specificky pro zařízení SMI REM-250 (v optimálním případě by samozřejmě byla aplikace univerzální pro všechny eye trackingová zařízení, to však není technicky možné – v kódu jsou volány specifické funkce definované v SDK (Software Development Kit) daného zařízení)
- aplikace bude řešena s důrazem na funkčnost a komplexnost serverové části tak, aby v budoucnu mohl být implementován zbytek klientské části a aplikace mohla být jednoduše propojena se softwarem Hypothesis

5 IMPLEMENTACE

5.1 Výběr nástrojů

5.1.1 SMI RED 250

Konkrétní eye trackingové zařízení bylo vybráno spolu se specifikacemi požadavků při samotném zadání a vychází z faktu, že daným zařízením je vybavena laboratoř HUME Lab II na Masarykově univerzitě. Jedná se o mobilní eye tracker, který je připojitelný na PC nebo notebook a běží na frekvenci 60, 120 a 250 Hz. Výrobcem uváděná přesnost je 0,4°, rozsah pohybu hlavy 40 cm x 20 cm a provozní vzdálenost 60 až 80 cm. Připojení je možné pouze přes USB kabel s portem 2.0. K samotnému spuštění zařízení je nezbytná instalace softwaru SMI iView X, který zajišťuje běh zařízení.

Dalším důvodem výběru tohoto zařízení je fakt, že disponuje velmi robustním SDK, jehož součástí je iView X API, které usnadňuje komunikaci mezi vlastní eye trackingovou aplikací a SMI eye trackingovým zařízením. Součástí SDK jsou také příklady kódů v různých programovacích jazycích, které demonstrují funkcionality, které API dává k dispozici [39].

5.1.2 Python 2

iView X SDK je možné využít s většinou programovacích a skriptovacích jazyků, které dokáží importovat DLL (dynamicky linkované knihovny), což jsou například jazyky C/C++, C# nebo Python. Pro tvorbu aplikace byl vybrán poslední jmenovaný, a to především z důvodu předchozí praktické zkušenosti s daným jazykem.

Prvotní vývoj aplikace probíhal v (dané době) nejnovější verzi jazyka 3.8.2, avšak během testování aplikace vyšlo najevo, že některé z využívaných knihoven jsou psány ve verzi 2 a chybí podpora verze 3, aplikace proto musela být přepsána do starší verze 2.7. Obě verze se však neliší nijak dramaticky, nejviditelnější rozdíly jsou ve funkci print, ve způsobu dělení celými čísly, v podpoře Unicode a ve způsobu ošetřování výjimek.

5.1.3 WebSockets

WebSockets je komunikační protokol na úrovni aplikace, který na rozdíl od HTTP protokolu umožňuje plně duplexní (oboustrannou) komunikaci. Klasická HTTP komunikace probíhá metodou, kdy klient iniciuje komunikaci odesláním požadavku (request) na server, který jej

zpracuje a následně odešle uživateli odpověď (response). Jedná se o bezstavovou komunikaci, což znamená, že ani klient ani server neukládají žádné informace o komunikaci a po vyřízení požadavků je spojení ukončeno. Pro vyřízení dalšího požadavku musí být spojení opět otevřeno. Oproti tomu WebSockets je komunikace v reálném čase navržená tak, aby po vyřízení požadavku zůstalo spojení navázáno a komunikace mohla oběma směry probíhat dál. Z důvodu zachování funkčnosti ve stávajícím prostředí webových aplikací je komunikace inicializována pomocí HTTP požadavku. Další výhodou WebSockets je fakt, že data oběma směry mohou být posílána jak ve formátu obyčejného textu, tak ve formě binárních dat. Díky navázání na protokol TCP je zajištěno, že vyměněná data od klienta i od serveru, dorazí neporušena a ve stejném pořadí.

5.2 Příprava prostředí

Před samotným vývojem bylo třeba nainstalovat Python 2.7 a SMI iView X SDK. K praktickému ověření funkčnosti kódu je nezbytné mít připojené eye trackingové zařízení, které běží pouze na operačním systému Windows s USB portem 2.0. Z tohoto důvodu je doporučováno provádět i samotný vývoj na zařízení s těmito parametry. Takové zařízení však bohužel nebylo autorce práce k dispozici, vývoj proto probíhal na zařízení s operačním systémem Linux Ubuntu 18.04 a testování probíhalo odděleně na dalším zařízení.

Následující příkazy nainstalují Python 2.7 (Obrázek 14):

```
sudo apt update
sudo apt upgrade
sudo apt install python2.7
sudo apt install python-pip
```

Obrázek 14: Ukázka kódu - instalace Pythonu 2.7

SMI iView X SDK není volně dostupný software. Firma SensoMotoric Instruments sice již neexistuje, ale podpora je stále ještě dostupná. Je na ni vyčleněn Gaze Intelligence tým, který byl součástí původního SMI [40]. Pro získání SDK je třeba emailem kontaktovat Gaze Intelligence. Uživatel je vyzván, aby uvedl ID konkrétního využívaného zařízení a na jeho základě je mu zaslán instalační soubor SDK. Na zařízení Windows stačí tento soubor spustit, na platformě Linux je nezbytné provést nejdříve instalaci programu Wine a následně pomocí Wine soubor spustit. Je možné využít následující příkazy (Obrázek 15):

```
sudo apt install wine64
wine iViewXSDK.exe
```

Obrázek 15: Ukázka kódu – instalace programu Wine

Dále je třeba mít nainstalované IDE podle vlastního uvážení.

5.3 Popis aplikace

5.3.1 Zajištění komunikace klient – server

5.3.1.1 Knihovna Socket

Obousměrná komunikace v reálním čase mezi klientem a serverem je zajišťována protokolem WebSockets. Implementace musí být zajištěna jak na straně serveru, tak na straně klienta, aby mohlo být navázáno spojení. Existuje několik způsobů implementace WebSockets v Pythonu 2, a to například:

- Autobahn
- Django Channels
- Flask-SocketIO
- Crossbar.io
- knihovna Socket

V projektu byla využita poslední zmíněná varianta, nízko-úrovňová knihovna socket, která je součástí standardní Python knihovny, a není proto třeba ji zvlášť instalovat.

Po importování knihovny byl definován socket objekt, opět na straně serveru i na straně klienta (Obrázek 16Obrázek 17). AF_INET určuje rodinu protokolů, v tomto případě IPv4, SOCK_STREAM pak definuje vlastní typ socketu, konkrétně TCP. Na serverové straně je třeba socket pomocí funkce *bind* svázat s n-ticí (tuple), kdy prvky budou IP adresa a port. Server poběží lokálně, číslo portu bylo vybráno tak, aby se eliminovalo riziko, že je již obsazen. Z toho důvodu jsou oba prvky definovány jako globální konstanty a společně uloženy do konstanty ADDR.

```
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDR)
```

Obrázek 16: Ukázka kódu – vytvoření objektu socket na straně serveru

Na straně klienta bude využit totožný socket se stejnými parametry, jediný rozdíl bude v tom, že místo svázání se bude socket pomocí funkce *connect* připojovat k serveru. Hodnoty globálních proměnných IP adresa a port budou v tomto případě totožné, neboť klient poběží lokálně, lze však nastavit i jiné parametry a k serveru se připojit vzdáleně. Právě této varianty bude využito v budoucnu při implementaci propojení se softwarem Hypothesis.

```
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(ADDR)
```

Obrázek 17: Ukázka kódu – vytvoření objektu socket na straně klienta

Kromě samotné definice protokolu WebSockets popsané výše je třeba vyřešit odesílání dat a jejich velikost, přijímání dat a jejich velikost, formát posílaných zpráv, způsob připojování dalších klientů a ošetření odpojení klientů. Řešení všech výše popsaných problémů bude vysvětleno v dalších kapitolách, protože se více týkají jednotlivých částí aplikace než samotného protokolu WebSockets.

5.3.1.2 Třída iViewXTracker

Třída iViewXTracker (Obrázek 18) obsahuje všechny důležité atributy a metody, které jsou využívané pro komunikaci mezi klientem a serverem. Některé z funkcí a datových struktur jsou definovány přímo v samotném API, které je součástí iViewX SKD. Pro přístup k nim je třeba importovat soubory iViewXAPI a iViewXReturnCodes.

Při samotné inicializaci třídy pomocí metody *__init__* dojde k načtení DLL knihovny do paměťového prostoru aplikace, odkud jsou následně volány předdefinované funkce a datové typy.

```
class iViewXTracker():

    writter = None

    def __init__(self):
        self.iViewXAPI = windll.LoadLibrary("iViewXAPI.dll")
        CMPFUNC = WINFUNCTYPE(c_int, CSample)
        self.smp_func = CMPFUNC(self.SampleCallback)
        self.sampleCB = False

        CMPFUNC = WINFUNCTYPE(c_int, CEvent)
        self.event_func = CMPFUNC(self.EventCallback)
        self.eventCB = False

    def connect(self):
        ret = self.iViewXAPI.iV_SetLogger(c_int(1), c_char_p("iViewX_experiment.txt"))
        ret = self.iViewXAPI.iV_Connect(c_char_p('127.0.0.1'), c_int(4444), c_char_p('127.0.0.1'), c_int(5555))
        if ret != 1:
            HandleError(ret)
            exit(0)

    def calibrate(self):
        self.iViewXAPI.iV_Calibrate()

    def validate(self):
        self.iViewXAPI.iV_Validate()
```

Obrázek 18: Ukázka kódu – třída iViewXTracker

Dále je součástí třídy metoda *connect*, která zabezpečuje vytvoření dokumentu pro logování případných chyb a navázání spojení s iView eye trackingovým serverem pomocí funkcí předdefinovaných v SDK *iV_SetLogger* a *iV_Connect*. Funkce *iV_Connect* obsahuje parametry IP adres iViex X počítače a lokálního počítače (v našem případě se jedná o totožnou lokální adresu) a čísla portu, které iView X SDK využívá k odesílání a přijímání dat z iView X. Funkce nevrátí hodnotu, dokud není navázáno spojení. V případě neúspěšného pokusu o spojení je volána funkce *HandleError*, která podle návratových hodnot vypíše vhodnou chybovou hlášku a následně celý program ukončí.

Metoda *calibrate* spustí kalibraci. Kalibrace je nezbytnou součástí procesu záznamu očních pohybů, protože díky ní je přístroj schopen stanovit fyziologická specifika očí daného účastníka výzkumu a optimalizovat tak přesnost měření. Typicky je kalibrací myšlena postupná prezentace bodů (jejich minimální počet je 5), na které účastník upírá svůj zrak. Tyto kalibrační body je následně nutné přijmout, což může být provedeno samotným účastníkem, avšak doporučovaným (a v aplikaci využitým) způsobem je automatické přijímání kalibračních bodů. V tomto módu server předpokládá, že účastník je ochotný spolupracovat a během kalibrace opravdu zaměřuje svůj zrak na kalibrační body.

Pro zhodnocení kvality provedené kalibrace je možné provést validaci zavoláním metody *validate*. Ta voláním funkce *iV_Validate* podobně jako u kalibrace spustí prezentaci 4 bodů.

Následně funkce *validate* vypočítá rozdíly mezi prezentovanými body a naměřenými body kalibrace.

Metoda *createFile* přiřadí do atributu writter instanci třídy FileWritter, která může být následně použita k zapisování dat do souboru a jejím parametrem je jméno souboru. Třídě FileWritter je věnována kapitola **Chyba! Nenašiel sa žiaden zdroj odkazov.**.

Metoda *startRecordingData* se používá k zápisu dat z eye trackeru do již vytvořeného souboru. K tomuto účelu funkce volá metody *iV_SetSampleCallback* a *iV_SetSampleCallback*, jejichž parametry jsou funkce *SampleCallback* a *EventCallback* určené k získávání a zápisu dat do souboru (Obrázek 19).

Metoda *disconnect* pomocí funkce *iV_Disconnect* ukončí spojení eye trackeru se serverem. Následně provede uzavření souboru pro zápis dat, pokud soubor ještě nebyl uzavřen.

```
def SampleCallback(self, sample):
    self.writter.writeRow([sample.timestamp, sample.leftEye.gazeX, sample.leftEye.gazeY, sample.rightEye.gazeX, sample.rightEye.gazeY])
    return 0

def EventCallback(self, event):
    return 0

def startRecordingData(self):
    res = self.iViewXAPI.iV_SetSampleCallback(self.smp_func)
    self.sampleCB = True
    res = self.iViewXAPI.iV_SetEventCallback(self.event_func)
    self.eventCB = True
```

Obrázek 19: Ukázka kódu – *callback* funkce

5.3.2 Vytvoření .csv souboru – třída FileWritter

Třída FileWritter (Obrázek 20) a její metody slouží k vytvoření .csv souboru, do kterého jsou ukládána data během nahrávání. Při inicializaci třídy dojde k vytvoření souboru, který má definovanou hlavičku v podobě časové známky počítače, souřadnice X levého oka, souřadnice Y levého oka, souřadnice X pravého oka a souřadnice Y pravého oka. Metoda *writeRow* zajišťuje zápis dat vždy na nový řádek, metoda *close* zavírá soubor po dokončení nahrávání a metoda *isClosed* kontroluje, zda je soubor uzavřen a ukončení programu tak nezpůsobí chybu z důvodu otevřeného souboru.

```python
import csv


class FileWritter():
    HEADERS = ["ComputerClock_Timestamp", "LeftEye_GazeX", "LeftEye_GazeY", "RightEye_GazeX", "RightEye_GazeY"]
    DELIMITER = ','

    def __init__(self, file_name):
        self.open_file = open(file_name, mode='w')
        self.writter = csv.writer(self.open_file, delimiter = FileWritter.DELIMITER)
        self.writter.writerow(FileWritter.HEADERS)

    def writeRow(self, data):
        self.writter.writerow(data)

    def close(self):
        self.open_file.close()

    def isClosed(self):
        return self.open_file.closed
```

Obrázek 20: Ukázka kódu – třída FileWritter

5.3.3 Běh funkcí ve vláknech – třída CustomThread

Na straně serverového kódu je třeba spouštět některé funkce ve vláknech, aby na sobě nebyly závislé a mohly dobíhat v různých časových úsecích. K tomuto účelu byla vytvořena třída CustomThread, která obsahuje kromě samotné inicializace jen dvě metody, *run* a *stop*. Instance této třídy jsou následně schopny upravovat běh vláken podle aktuálních potřeb. Ukázka dané třídy na Obrázek 21.

```python
import threading

class CustomThread(threading.Thread):

    def __init__(self, tracker):
        threading.Thread.__init__(self)
        self.tracker = tracker

    def run(self):
        self.tracker.startRecordingData()
        print '[RECORDING...]'


    def stop(self):
        self.run = False
```

Obrázek 21: Ukázka kódu – třída CustomThread pro běh funkcí ve vláknech

5.3.4 Server

V úvodu kódu na straně serveru jsou importovány knihovny socket, datetime a threading a třídy iViewXTracker a CustomThread (Obrázek 22Obrázek 22). Definovány jsou též parametry objektu socket, které jsou blíže popsány v kapitole 5.3.1.1.

```
import socket
from iViewXClasses import iViewXTracker
from customThread import CustomThread
from datetime import datetime
import threading
```

Obrázek 22: Ukázka kódu – import knihoven a tříd

Dále jsou globálně definované konstanty HEADER, PORT, SERVER, ADDR, FORMAT a DISCONNECT_MESSAGE. Konstanta HEADER určuje velikost headeru zprávy, kterou posíláme první a kódujeme do ní informaci o tom, jak velká bude následná zpráva. Konstanty PORT a SERVER určují, na kterém portu a na jaké IP adrese server poběží. Konstanta ADDR svazuje konstanty PORT a SERVER do n-tice tak, aby je mohla dále využívat funkce *bind* při definování protokolu WebSockets. Konstanta FORMAT určuje, že dekódovaná zpráva je ve formátu utf-8. Poslední konstantou je DISCONNECT_MESSAGE, do které je uložena zpráva, při které se spojení ukončí.

```
HEADER = 128
PORT = 5050
SERVER = '127.0.0.1'
ADDR = (SERVER, PORT)
FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "Disconnected!"
```

Obrázek 23: Ukázka kódu – globálně definované konstanty

Na straně serveru jsou definovány dvě nejdůležitější funkce, a to *handle_client* a *start*. Funkce *start* (ukázka kódu na Obrázek 24) umožňuje serveru naslouchat a očekávat nově příchozí spojení a následně tato nová spojení předat funkci *handle_client*. Obě funkce běží v separátních vláknech, jejichž běh je umožněn díky importované knihovně threading (využitá technologie WebSockets umožňuje připojení vícero klientů zároveň, v případě daného kódu se jedná o 5 klientů. Z praktického hlediska to aktuálně není scénář, který by zadavatel využil, funkcionalita je však zachována pro případ změn v budoucnu).

V konstantě CONN je uložen socket objekt, v konstantě ADDR je uložena IP adresa nově připojeného klienta a dané proměnné jsou metodou *server.accept* přijaty jako nové spojení. Nově navázané spojení je následně předáno funkci *handle_client*, parametry jsou opět proměnné conn a addr. Na závěr funkce vypíše počet připojených klientů ve vláknech.

```
def start():
    server.listen(5)
    print "[LISTENING] Server is listening on " + str(SERVER)
    while True:
        conn, addr = server.accept()
        handle_client(conn, addr)
        print "[ACTIVE CONNECTIONS] " + str(threading.activeCount() - 1)


print("[STARTING] server is starting...")
start()
```

Obrázek 24: Ukázka kódu – funkce *start*

V okamžiku, kdy se k serveru připojí klient a je tedy splněna podmínka uvnitř funkce *handle_client*, server čeká na zprávy od klienta. Nejprve server obdrží zprávu s pevnou délkou (HEADER), ve které je zakódována velikost následující zprávy, kterou očekává. Následně je velikost dekódována a přijata zpráva s již známou velikostí s informací, jakou akci má server provést. Ukázka kódu funkce *handle_client* je na Obrázek 25.

Pokud se jedná o zprávu „connect tracker“, je vytvořena instance třídy iViewXTracker pojmenovaná tracker, přes kterou lze eye tracker ovládat. Dále instance třídy zavolá metodu *connect*, připojí tracker k eye trackingovému serveru a zároveň odešle klientovi zprávu, že eye tracker je připravený k použití.

Obdobným způsobem jsou řešeny funkce *calibrate* a *validate* – pokud server obdrží od klienta danou zprávu, spustí kalibraci, respektive validaci a následně odešle zprávu zpět klientovi, že dané akce proběhly.

```
def handle_client(conn, addr):
    connected = True
    while connected:
        msg_length = conn.recv(HEADER).decode(FORMAT)
        if msg_length:
            msg_length = int(msg_length)
            msg = conn.recv(msg_length).decode(FORMAT)
            if(msg == 'connect tracker'):
                tracker = iViewXTracker()
                tracker.connect()
                conn.send("Tracker API loaded. Eye-tracker is ready for calibration.".encode(FORMAT))
            if(msg == 'calibrate'):
                tracker.calibrate()
                conn.send("Tracker has been calibrated".encode(FORMAT))
            if(msg == 'validate'):
                tracker.validate()
                conn.send("Calibration has been validated".encode(FORMAT))
```

Obrázek 25: Ukázka kódu – *funkce handle_client*

V případě, že server obdrží od klienta zprávu ‚record', proběhne vícero akcí. Funkce volá metodu *createFile* na instanci třídy iViewXTracker, která má parametr file_name. Ten obsahuje definici podoby názvu vytvářeného souboru, který kromě slova ‚eyetracking' obsahuje i aktuální čas ve formátu rok-měsíc-den-hodina-minuta-vteřina. Specifické určení času je nezbytné z důvodu, že v jeden den může probíhat větší množství výzkumů a je proto nutné mít každý vytvořený soubor specificky pojmenovaný, aby nedošlo k záměně testovacích dat. Daný soubor je uložen do stejné složky, ve které se nacházejí soubory s kódem. Funkce posílá klientovi zprávu informující jej o tom, do jakého souboru budou nahrávaná data ukládána. Aby mohlo být později nahrávání dat ukončeno bez nutnosti čekat, až akce doběhne (což by způsobilo pád aplikace, protože nahrávání je zacykleno podmínkou), je vytvořena instance třídy CustomThread pojmenovaná data_thread a následně je na ni volána metoda *start*, která spustí nahrávání dat v separátním vlákně. Pokud je přijata zpráva ‚stop record', ukončí se nahrávání a vlákna se spojí, aby běh aplikace pokračoval již v jednom vlákně. Následně je odpojen eye tracker od eye trackingového serveru a klientovi je odeslána zpráva, že data byla úspěšně uložena a eye tracker byl odpojen. Po doběhnutí celé funkce *handle_client* je ukončené WebSockets spojení mezi klientem a WebSockets serverem, který však běží dál. Je tak možné se k němu opětovně připojit.

V kódu je zapracovaná i podmínka pro neočekávané odpojení eye trackeru od serveru. V takovém případě je klientovi odeslána zpráva, že eye tracker byl odpojen.

5.3.5 Klient

Stejně jako serverová část aplikace ani klientská část neřeší UI, neboť v konečné podobě bude na straně klienta webová aplikace Hypothesis, která bude současně sloužit jako frontend pro veškerou uživatelskou interakci. Všechna komunikace se serverem je v současné podobě tedy zajišťována přes příkazový řádek.

Ze zadání vyplynulo, že důraz při implementaci má být kladen na serverovou část aplikace a definování způsobu komunikace mezi serverem a klientem. Z principu věci však nemohla být klientská část vynechána úplně, obsahuje tedy alespoň kostru toho, jak bude časem pokračovat implementace.

Jedinou implementovanou knihovnou je socket, globálně definované konstanty jsou totožné jako v případě serveru, neboť jsou využívány na obou místech. Následuje nastavení protokolu WebSockets, blíže popsané v kapitole 5.3.1.1. Pro komunikaci se serverem je definovaná funkce *send* s parametrem msg (Obrázek 26). Do proměnné message je přiřazen zakódovaný řetězec msg, protože zprávy budou přes WebSockets odesílány v bajtech. Protože je potřeba následovat protokol, je nejprve poslána zpráva s pevně definovanou velikostí (HEADER), která v sobě obsahuje informaci o velikosti následující zprávy. Následně je odeslána daná zpráva reálné velikosti a klient čeká na odpověď ze serveru. Odpověď serveru je aktuálně vypisována do konzole, v budoucnu však může takhle část být upravena tak, aby bylo možno odesílat soubor s nahranými daty.

```
def send(msg):
    message = msg.encode(FORMAT)
    msg_length = len(message)
    send_length = str(msg_length).encode(FORMAT)
    send_length += b' ' * (HEADER - len(send_length))
    client.send(send_length)
    client.send(message)
    print(client.recv(4096).decode(FORMAT))
```

Obrázek 26: Ukázka kódu – funkce *send*

Poslední část kódu zajišťuje ovládání eye trackeru ze strany uživatele přes příkazový řádek. Pokud je splněna podmínka, že klient je připojen, může si vybírat z definované nabídky příkazů. Ty jsou totožné funkcím na straně serveru: pomocí tlačítka ‚c‘ je možné připojit eye tracker k serveru, tlačítka ‚a‘ a ‚v‘ zajištují kalibraci a validaci, tlačítko ‚r‘ spouští nahrávání dat a tlačítko ‚s‘ ukončuje nahrávání. Odpojit eye tracker je možné pomocí příkazu ‚quit‘.

Na rozdíl od ostatních příkazů není definován písmenem, ale slovem, aby se předešlo nechtěnému odpojení z důvodu „ukliknutí“ na klávesnici. Pokud uživatel stiskne jinou klávesu, je mu zobrazeno upozornění, že dané tlačítko nic neovládá.

```
connected = True
while connected:
    user_input =  raw_input("What action do you want to do now? You can connect (c), calibrate (a), validate (v), \
        start recording (r), stop recording (s) or disconnect (quit): ")
    if user_input=="c":
        send("connect tracker")
    elif user_input=="a":
        send("calibrate")
    elif user_input=="v":
        send("validate")
    elif user_input=="r":
        send("record")
    elif user_input=="s":
        connected = False
        send("stop record")
    elif user_input=="quit":
        connected = False
        send(DISCONNECT_MESSAGE)
    else:
        print "Unsupported action, you can only connect (c), calibrate (a), validate (v), \
            start recording (r), stop recording (s) or disconnect (quit)"
```

Obrázek 27: Ukázka kódu – ovládání eye trackeru ze strany uživatele

5.4 Návrh řešení komunikace se softwarem Hypothesis

Jak již bylo zmíněno, aplikace aktuálně není propojena se softwarem Hypothesis, neboť nebyl zajištěn přístup ke zdrojovému kódu, bez kterého nelze provést potřebné změny. Tento fakt byl zjištěn během vývoje aplikace a byl otevřeně komunikován zadavateli. Reakcí byla úprava zadání a změna specifikace požadavků tak, aby bylo možné vytvořit alespoň předstupeň konečné podoby aplikace a na ni bylo možno navázat při dalším vývoji samotným autorem softwaru Hypothesis. Daný předstupeň je aplikace v takové podobě, v jaké je popsaná v této práci.

Cílem této kapitoly je nastínit přibližný proces při další implementaci, aby mohla vzniknout plně využitelná aplikace. Z praktického hlediska je implementováno navázání spojení a posílání zpráv přes protokol WebSockets, a to jak na straně klienta, tak i na straně serveru. Celé řešení je připraveno tak, aby po implementaci nezbytných změn na klientské straně aplikace mohla být tato plně využívána. Konkrétně musí být implementována JavaScript knihovna pro WebSockets, například AutobahnJS; pro usnadnění implementace je možné využít některý z wrapperů pro usnadnění volání knihovny, neboť serverová část je psaná v Pythonu. Dále musí být na straně klienta upraveno nastavení n-tice s prvky IP adresy a portu podle toho, na jaké adrese poběží Hypothesis.

Následně je třeba data z nahrávání uložená do textového souboru zakódovat pomocí modulu json, jehož metoda *dumps* uloží data z textového souboru, na straně klienta pak stejný modul a metoda *loads* data načte.

Poslední nezbytnou úpravou je implementace uživatelského rozhraní, pomocí kterého bude možné jednoduše a intuitivně eye tracker ovládat.

6 TESTOVÁNÍ

Tato kapitola se věnuje testování aplikace v reálných podmínkách a závěrečnému zhodnocení funkčnosti aplikace a splnění požadavků a zadání koncovými uživateli, potažmo zadavateli.

6.1 Testovací prostředí

Aplikace nemohla být testována během samotného vývoje, neboť vývoj probíhal na zařízení s operačním systémem Linux a USB portem 3.0 z důvodu, že autorka práce neměla k dispozici zařízení s USB portem 2.0. Technické požadavky ze samotného principu věci (eye trackingové zařízení lze spustit pouze na operačním systému Windows a připojením do USB portu 2.0) však vyžadují provozu schopnost aplikace s výše zmíněnými požadavky na zařízení. Tyhle parametry bohužel nelze obejít instalací virtuálního prostředí Windows s USB 2.0 kontrolorem, jedná se o hardwarové omezení.

Další limitací byl fakt, že eye trackingové zařízení je umístěno v laboratoři Masarykovy univerzity a nelze jej volně vypůjčit domů. Testovacím zařízením proto byl notebook od společnosti SMI, který je součástí pracovní stanice iView X.

Před samotným testováním bylo proto třeba na testovací zařízení nainstalovat následující nástroje (více informací o nastavení prostředí v kapitole 5.2):

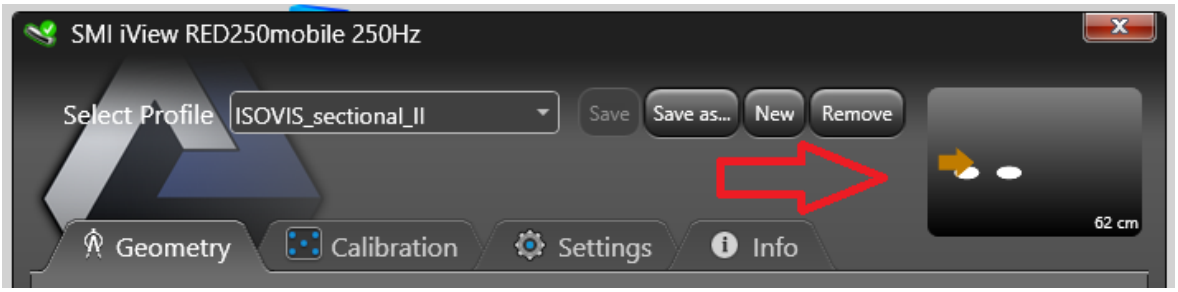
- Python 2
- SMI iView X SDK

Již nainstalovaný byl iView eye tracking server, v jiném případě by bylo třeba doinstalovat i ten.

6.2 Ověření funkčnosti aplikace

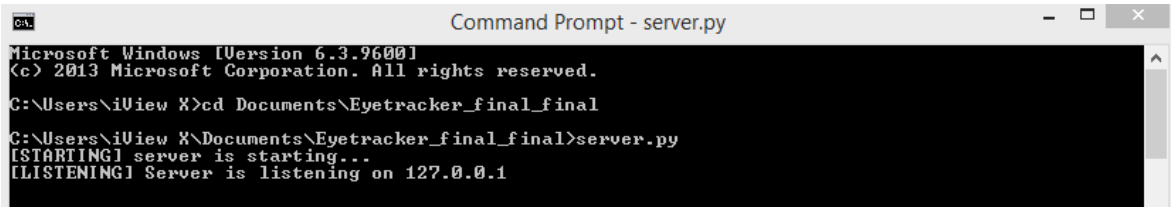
Po instalaci a nastavení všech potřebných nástrojů bylo pomocí USB 2.0 portu připojeno eye trackingové zařízení. Na monitoru testovacího zařízení je umístěn kovový obdélník, který usnadňuje upevnění eye trackeru k notebooku. Při jeho upevnění pomocí magnetu je však třeba dbát na to, aby bylo zařízení otočeno správným směrem. Zařízení neposkytuje žádné vodítko pro způsob umístění, jedinou zpětnou vazbou pro uživatele, že zařízení není upevněno správně, je tak pouze špatná detekce očí. Následně byl spuštěn iView X eye tracking server. Na jeho straně je vždy třeba ověřit, že oči jsou správně detekovány (příklad špatného

postavení očí na Obrázek 28). K tomuto účelu slouží jednoduché uživatelské rozhraní na straně eye trackingového serveru iView X, které zobrazuje, zda a v jaké pozici jsou oči detekovány a kterým směrem je případně třeba se posunout, aby byly oči bez problému detekovatelné v průběhu celého experimentu. Umístění očí lze kromě upravení vzdálenosti testované osoby (ideální vzdálenost od monitoru je cca 60 centimetrů) také poměrně dobře regulovat naklopením monitoru notebooku.



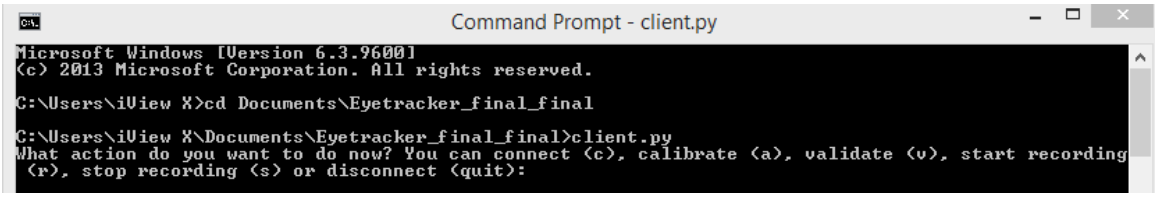
Obrázek 28: Uživatelské rozhraní eye trackingového serveru iView X

Po splnění všech předešlých kroků je možné spustit samotný kód. K tomu je třeba si otevřít dva terminály, jeden pro spuštění souboru server.py a druhý pro následné spuštění souboru client.py. Jak již bylo zmíněno, součástí aplikace není UI, protože veškerá interakce uživatele s aplikací bude ve své finální podobě probíhat přes software Hypothesis. Po spuštění souboru server.py je uživatel informován o tom, že server byl spuštěn a na jaké IP adrese běží (Obrázek 29). Až do okamžiku uložení nahraných dat je jeho běh nepřetržitý.



Obrázek 29: Spuštění souboru server.py – uživatel je informován o tom, zda a na jaké adrese server běží

V druhém terminálu je následně spuštěn soubor client.py. Po jeho spuštění se uživateli zobrazí informace, jakým způsobem může ovládat eye trackingové zařízení a jaké akce má k tomu dispozici (Obrázek 30).

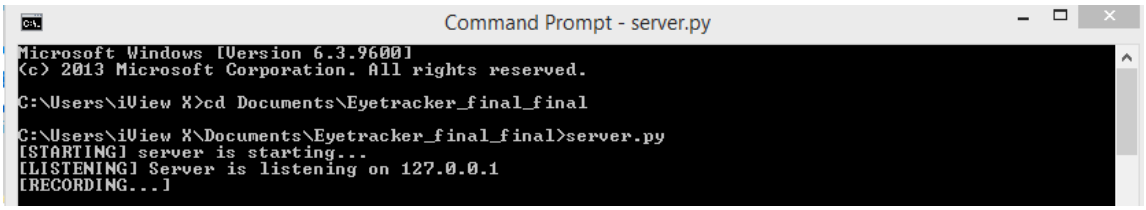


Obrázek 30: Spuštění souboru client.py – uživatel je informován o tom, jakými způsoby může ovládat eye trackingové zařízení

V případě stisknutí klávesy ‚c' je eye tracker připojen k serveru, klávesa ‚a' zajišťuje spuštění automatické pěti bodové kalibrace (ukázka na Obrázek 31). Ta má podobu šedé plochy přes celou obrazovku, kdy je uživateli prezentován červený bod s bílým ohraničením, na který musí upřít svoji pozornost. V okamžiku, kdy to provede a eye tracker získá fyziologická data o pohledu oka, posune se bod na další místo a celý proces je opakován až do úspěšného proběhnutí kalibrace. Následně může (avšak nemusí) být klávesou ‚v' spuštěna validace dat získaných při kalibraci. Vizuálně má stejnou podobu, pouze není prezentováno 5 fixačních bodů, nýbrž 4.

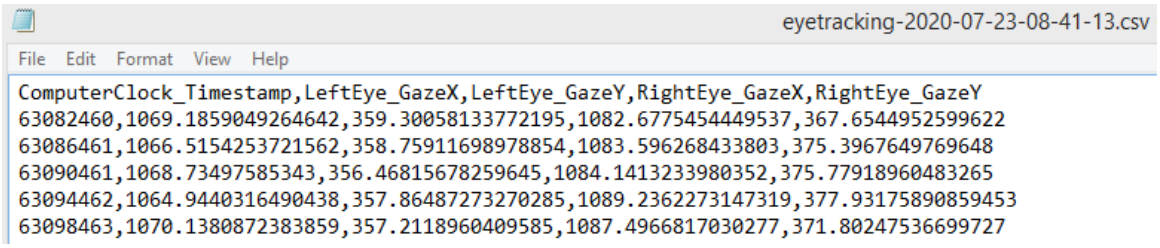
Obrázek 31: Ukázka obrazovky po spuštění kalibrace (případně i validace): vlevo nahoře lze vidět kalibrační bod (červená tečka s bílým ohraničením)

Po stisknutí klávesy ‚r' je spuštěno nahrávání dat o pohybu očí. Informace o tom, že nahrávání probíhá se zobrazí v terminálu serveru (Obrázek 32), na straně klienta je však tato část aktuálně bez vizuální odezvy. V případě ostrého provozu aplikace bude v téhle pasáži uživateli prezentován experimentální podnětový materiál.



Obrázek 32: Informace serveru o tom, že probíhá nahrávání dat

Po stisknutí klávesy ‚s' dojde k ukončení nahrávání, uložení souboru s nahranými daty a odpojení eye trackeru (ukázka formátu uložených dat je na Obrázek 33). O všech zmíněných akcích je uživatel informován. WebSockets server běží dál a klient se k němu může opětovně připojit.



Obrázek 33: Uložená data jsou ve formátu časová známka, souřadnice X a Y levého oka a souřadnice X a Y pravého oka

V Příloze P1 lze najít přesný přepis video záznamu pořízeného během funkčního testování aplikace. Samotný video záznam je součástí elektronických příloh této práce.

6.3 Přenositelnost kódu

I přesto, že byl kód primárně vyvíjen na platformě Linux, přenositelnost kódu bohužel není zachována. Kód musel být během testování upraven tak, aby byl funkční na zařízení s operačním systémem Windows, hlavním důvodem je fakt, že při samotné inicializaci SDK je třeba načíst iViewXAPI.dll, což je dynamicky linkovaná knihovna společnosti Microsoft. Dalším omezením je i samotná verze operačního systému, konkrétně Windows XP/7/8, neboť eye trackingové zařízení je podporována pouze na zmíněných verzích.

Zároveň není možné aplikaci využít k ovládání jiného eye trackingového zařízení než SMI RED-250, protože v aplikaci jsou volány specifické funkce, jež jsou definovány v samotném SDK.

Při nainstalování všech potřebných nástrojů zmíněných v kapitole 6.1 a vybrání hardwarově vhodného zařízení lze aplikaci spustit na jakémkoliv zařízení s platformou Windows XP/7 nebo 8.

6.4 Zpětná vazba uživatelů

Během demonstrace kódu byla zadavateli předvedena funkčnost aplikace a její aktuální podoba. Předvedení aplikace se dále zúčastnili také výzkumníci z řad psychologů a kartografů, kteří budou aplikaci nejvíce využívat a pro které bude ve své konečné podobě představovat důležitý nástroj, který aktuálně nemají k dispozici. Na základě jejich zpětné vazby lze konstatovat, že definované požadavky byly zrealizovány, zadání splněno a aplikace má všechny požadované funkcionality. K okamžitému započetí využívání aplikace chybí implementace na straně Hypothesis, byla však vyjádřena naděje, že k implementaci by mohlo dojít v brzké době, protože autor platformy Hypothesis přislíbil uvolnění zdrojového kódu.

ZÁVĚR

Cílem práce bylo vytvořit aplikaci, pomocí které by bylo možné ovládat eye trackingové zařízení, a navrhnout případné propojení eye trackeru se softwarem Hypothesis. V teoretické části se práce zabývá popisem zrakového aparátu a očních pohybů, protože nezbytně souvisí s metodou sledování očí. Dále jsou popsány metody eye trackingu a jejich vývoje v čase. Další kapitola se věnuje originální výzkumné platformě Hypothesis, která vznikla na Masarykově univerzitě s přispěním Centra experimentální psychologie a kognitivních věd (CEPSoS), Přírodovědecké a Filozofické fakulty Masarykovy univerzity a firmy Tilioteo. Kromě použití ve výzkumu je software komerčně využíván ve Vojenské nemocnici v Brně a Ústřední vojenské nemocnici v Praze. V rámci této práce je také popsán princip fungování Hypothesis. Poslední, třetí kapitola představuje krátký úvod do problematiky softwarových řešení eye trackingu a popisu několika konkrétních aplikací.

V rámci práce vznikla funkční aplikace, která je schopná ovládat eye trackingové zařízení SMI RED-250. Konkrétně je aplikace schopna připojit eye tracker, spustit kalibraci zařízení a následnou validaci naměřených dat, spustit a ukončit nahrávání očních pohybů, uložení dat do souboru a odpojení eye trackeru. Zároveň aplikace díky použití technologie WebSockets umožňuje komunikaci s klientem. V aktuální podobě je na straně klienta příkazový řádek, v plánu je však pokračovat v rozšiřování aplikace tak, aby na straně klienta byla samotná platforma Hypothesis.

Práce obsahuje také potřebnou dokumentaci, která vysvětluje řešený problém i zadané požadavky, použité nástroje a jejich instalaci, přípravu vývojového i testovacího prostředí a popis zdrojového kódu.

SEZNAM POUŽITÉ LITERATURY

[1] DUCHOWSKI, Andrew. *Eye tracking methodology: theory and practice*. Third edition. Cham: Springer, 2017. ISBN 978-3-319-57881-1.

[2] IMotions: Eye Tracker Prices – An Overview of 20+ Eye Trackers [online*]. 2019 [cit. 2020-08-04]*. Dostupné z: https://imotions.com/blog/eye-tracker-prices/

[3] HOLMQVIST, Kenneth, Marcus NYSTRÖM, Richard ANDERSSON, Richard DEWHURST, Halszka JARODZKA a Joost WEIJER. *Eye tracking: a comprehensive guide to methods and measures*. First published. Oxford: Oxford University Press, 2011. ISBN 978-0-19-969708-3.

[4] POPELKA, Stanislav, *Zdeněk STACHOŇ, Čeněk ŠAŠINKA a Jitka DOLEŽALOVÁ. EyeTribe Tracker* Data Accuracy Evaluation and Its Interconnection with Hypothesis Software for Cartographic Purposes. *Computational Intelligence and Neuroscience* [online]. 2016, **2016,** 1-14 [cit. 2018-11-11]. DOI: 10.1155/2016/9172506. ISSN 1687-5265. Dostupné z: http://www.hindawi.com/journals/cin/2016/9172506/

[5] *BOJKO, Aga.* Eye tracking the user experience: a practical guide to research. Brooklyn, New York: Rosenfeld Media, 2013. ISBN 1457103044.

[6] Anatomie lidského oka *[online]. [cit. 2020-01-12]. Dostupné z: http://www.*optika-safarikova.cz/images/content/oko_schema_m.jpg

[7] POPELKA, Stanislav. Eye-tracking (nejen) v kognitivní kartografii: praktický průvodce tvorbou a vyhodnocením experimentu. 1. vydání*. Olomouc:* Univerzita Palackého v Olomouci **pro** katedru geoinformatiky, 2018. ISBN 978-80-244-5313-2.

[8] STRASBURGER, H.*, I.* RENTSCHLER a M. JUTTNER. Peripheral vision and pattern recognition: A review. Journal of Vision [online]. 2011, **11**(5), 13-13 [cit. 2020-01-12]. DOI: 10.1167/11.5.13. ISSN 1534-7362. Dostupné z: http://jov.arvojournals.org/Article.aspx?doi=10.1167/11.5.13

[9] *Zorné pole. Encyklopedie* fyziky [online]. [cit. 2020-01-19]. Dostupné z: http://fyzika.jreichl.com/main.article/view/488-zorne-pole

[10] Vývoj a stavba oka [online]. [cit. 2020-01-13]. *Dostupné z: http://www.cvikyprooci.cz/vyvoj-a-stavba-oka/*

[11] VON NOORDEN, Gunter K. *a E. C.* CAMPOS. Binocular vision and ocular motility: theory and management of strabismus. 6th ed. St. Louis, Mo.: Mosby, 2002. ISBN 0323011292.

[12] HUBEL, David H. Eye, brain, and vision. New York: Distributed by W.H. Freeman, 1988. ISBN *0716750201.*

[13] MARTINEZ-CONDE, S. a S. L. MACKNIK. Fixational eye movements across vertebrates: Comparative dynamics, physiology, and perception. Journal of Vision [online]. 2008, 8*(14)*, 28-28 [cit. **2020-01-14**]. DOI: 10.1167/8.14.28. ISSN 1534-7362. Dostupné z: http://jov.arvojournals.org/Article.aspx?doi=10.1167/8.14.28

[14] ZÉNON, Alexandre, Brian D. CORNEIL, Andrea ALAMIA, Nabil FILALI-SADOUK, Etienne OLIVIER a Joy J. GENG. *Counterproductive Effect of Saccadic Suppression during Attention Shifts. PLoS ONE [online*]. 2014, 9(1) [cit. 2020-01-19]. DOI: 10.1371/journal.pone.0086633. ISSN 1932-6203. Dostupné z: http://dx.plos.org/10.1371/journal.pone.0086633

[15] SALVUCCI, Dario D. a Joseph H. GOLDBERG. Identifying fixations and saccades in *eye-tracking* protocols. In: Proceedings **of** the symposium on Eye tracking research & applications - ETRA '00 [online]. New York, New York, USA: ACM Press, 2000, s. 71-78 [cit. 2018-11-17]. DOI: 10.1145/355017.355028. ISBN 1581132808. Dostupné z: http://portal.acm.org/citation.cfm?doid=355017.355028

[16] DODGE, Raymond a Thomas Sparks CLINE. The angle velocity of eye movements. Psychological *Review [online]. 1901,* 8(2), 145-157 [cit. 2018-11-25]. DOI: 10.1037/h0076100. ISSN 0033-295X. Dostupné z: http://content.apa.org/journals/rev/8/2/145

[17] YOUNG, Laurence R. *a David SHEENA. Survey of eye movement recording methods. Behavior Research Methods &* Instrumentation [online]. 1975, 7(5), 397-429 [cit. 2018-11-25]. DOI: 10.3758/BF03201553. ISSN 1554-351X. Dostupné z: http://www.springerlink.com/index/10.3758/BF03201553

[18] BUSWELL, Guy Thomas. How people *look at pictures:* a study of the psychology and perception in art. Chicago: University of Chicago Press Chicago, 1935.

[19] SYNEK, Svatopluk a Šárka SKORKOVSKÁ. Fyziologie oka a vidění. 2., dopl. a přeprac. vyd. Praha: Grada, 2014. ISBN 978-80-247-3992-2.

[20] *REDA, Radwa, Manal TANTAWI,* Howida SHEDEED a Mohamed F. TOLBA. Analyzing Electrooculography (EOG) for Eye Movement Detection. HASSANIEN, Aboul Ella, Ahmad Taher AZAR, Tarek GABER, Roheet BHATNAGAR a Mohamed F. TOLBA, ed. *The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2019)* [online]. Cham: Springer International Publishing, 2020, s. 179-189 [cit. 2020-01-26]. Advances in Intelligent Systems and Computing. DOI: 10.1007/978-3-030-14118-9_18. ISBN 978-3-030-14117-2. Dostupné z: http://link.springer.com/10.1007/978-3-030-14118-9_18

[21] MINDMEDIA: Neuro and biofeedback systems [online]. [cit. 2020-01-26].

[22] *WHITMIRE,* Eric, Laura TRUTOIU, Robert CAVIN, David PEREK, Brian SCALLY, James PHILLIPS a Shwetak PATEL. EyeContact. In: *Proceedings of the 2016 ACM International Symposium on Wearable Computers - ISWC '16* [online]. New York, New York, USA: ACM Press, 2016, s. 184-191 [cit. 2018-11-26]. DOI: 10.1145/2971763.2971771. ISBN 9781450344609. Dostupné z: http://dl.acm.org/citation.cfm?doid=2971763.2971771

[23] Rotation Matrices [online]. [cit. 2018-11-26]. Dostupné z: http://work.thaslwanter.at/Kinematics/html/03_RotMats.html

[24] *HANSEN, D.W. a QIANG* JI. In the Eye of the Beholder: A Survey of Models for Eyes and Gaze. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 2010, **32**(3), 478-500 [cit. 2020-01-31]. DOI: 10.1109/TPAMI.2009.30. ISSN 0162-8828. Dostupné z: http://ieeexplore.ieee.org/document/4770110/

[25] *Eye Tracking:* The Complete Pocket Guide [online]. [cit. 2020-01-31]. Dostupné z: https://imotions.com/blog/eye-tracking/

[26] IViewX System Manual. Version 2.*8*. 2014.

[27] Apple acquires SMI eye-tracking company. Techcrunch [online]. [cit. 2020-03-29]. Dostupné z: https*://techcrunch.*com/2017/06/26/**apple-acquires-smi-eye-tracking-company**/

[28] SOGO, Hiroyuki. *GazeParser:* an open-source and multiplatform library for low-cost eye tracking and analysis. Behavior Research Methods [online]. 2013, **45**(3), 684-695 [cit. 2020-07-03]. DOI: 10.3758/s13428-012-0286-x. ISSN 1554-3528. Dostupné z: http://link.springer.com/10.3758/s13428-012-0286-x

[29] GazeParser/SimpleGazeTracker. GazeParser/SimpleGazeTracker [online]. [cit. 2020-07-04]. Dostupné z: http://gazeparser.sourceforge.net/

[30] *DALMAIJER, Edwin S., Sebastiaan MATHÔT a Stefan* VAN DER STIGCHEL. PyGaze: An open-source, cross-platform toolbox for minimal-effort programming of eyetracking experiments. *Behavior Research Methods* [online]. 2014, **46**(4), 913-921 [cit. 2018-11-27]. DOI: 10.3758/s13428-013-0422-2. ISSN 1554-3528. Dostupné z: http://link.springer.com/10.3758/s13428-013-0422-2

[31] PyGaze: Open-source toolbox for eye tracking in Python [online]. [cit. 2018-11-27]. Dostupné z: http://www.pygaze.org/

[32] SAN AGUSTIN, Javier, Henrik SKOVSGAARD, Emilie MOLLENBACH, Maria BARRET, Martin TALL, Dan Witzner HANSEN a John Paulin HANSEN. Evaluation of a low-cost open-source gaze tracker. In: *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications - ETRA '10* [online]. New York, New York, USA: ACM Press, 2010, s. 77- [cit. 2020-08-02]. DOI: 10.1145/1743666.1743685. ISBN 9781605589947. Dostupné z: http://portal.acm.org/citation.cfm?doid=1743666.1743685

[33] ŠAŠINKA, Čeněk a Kamil MORONG. Původní výzkumný nástroj *pro oblast* kartografie a psychologie – Multivariantní Testovací program (MuTeP). In: HALAMA, Peter, Róbert HANÁK a Radomír MASARYK. *Sociálne procesy a osobnosť 2012: Zborník príspevkov z 15. ročníka medzinárodnej konferencie*. Bratislava: Ústav experimentálnej psychológie, 2012, s. 188-194. ISBN 978-80-88910-40-4.

[34] Testovací software vyvinutý psychology z Muni využijí vojáci. Zprávy z muni [online]. 2018 [cit. 2020-02-02]. Dostupné z: https://www.em.muni.cz/*veda-a-vyzkum/10470-testovaci-software-vyvinuty-psychology-z-muni-vyuziji-vojaci*

[35] ŠAŠINKA, Čeněk, Kamil MORONG a Zdeněk STACHOŇ. The Hypothesis Platform: An Online *Tool for Experimental Research into Work* with Maps and Behavior in Electronic Environments. ISPRS International Journal of Geo-Information [online]. 2017, 6(12) [cit. 2018-11-20]. DOI: 10.3390/ijgi6120407. ISSN 2220-9964. Dostupné z: http://www.mdpi.com/2220-9964/6/12/407

[36] ŠTĚRBA, Zbyněk, *Čeněk ŠAŠINKA, Zdeněk* STACHOŇ, Radim ŠTAMPACH a Kamil MORONG. *Selected Issues of Experimental Testing in Cartography* [online]. Brno: Masarykova univerzita, 2015 [cit. 2020-02-02]. DOI: 10.5817/CZ.MUNI.M210-7893-2015. ISBN 978-80-210-7893-2.

[37] *MORONG, Kamil. Hypothesis: Příklady užití.* Brno: Tilioteo Ltd., 2015.

[38] OGAMA - Open Gaze and Mouse Analyzer [online]. [cit. 2018-11-27]. Dostupné z: http://www.ogama.net

[39] *IView X SDK Manual*. 4.2. 2015.

[40] GAZE INTELLIGENCE - Solutions, Behavioral Studies. *GAZE INTELLIGENCE - Solutions, Behavioral Studies* [online]. [cit. 2020-07-05]. Dostupné z: https://gazeintelligence.com/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API Application Interface

ET Eye tracking

ID Identifikace

IDE Integrated Development Environment (vývojové prostředí)

RED Remote Eyetracking Device

SDK Software Development Kit

SMI SensoMotoric Instruments

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P 1: PŘEPIS VIDEO ZÁZNAMU PREZENTACE FUNKČNOSTI APLIKACE

[0:00] Na začátku videa lze vidět již otevřené a předchystané dva příkazové řádky

[0:03] Je spuštěn iView server, bez jehož spuštění není možné eye tracker využívat

[0:20] V levém příkazovém řádku je spuštěn soubor server.py, který spustí WebSockets server

[0:25] Uživatel je informován o tom, že server byl spuštěn a na jaké IP adrese běží.

[0:28] V pravém příkazovém řádku je spuštěn soubor client.py. Ten se uživatele zeptá, jaké akce chce provést. Následně jsou postupně zadány nabízené akce.

[0:33] Stisknutím kláves ‘c’ a ‘enter’ eye tracker připojen připravena na kalibraci

[0:37] Po stisknutí klávesy ‘a’ a ‘enter’ je spuštěna kalibrace. Ta se uživateli zobrazí jako šedá obrazovka s červeným bodem s bílým okrajem. Po fixaci pohledu na bod se tento čtyřikrát přemístí.

[0:52] Stisknutím kláves ‘v’ a ‘enter’ spuštěna kalibrace. Uživatel vidí totožnou obrazovku, bod se tentokrát přemístí třikrát.

[1:06] Stisknutím kláves ‘r’ a ‘enter’ je spuštěno nahrávání pohybů očí. Uživatel je informován v příkazovém řádku serveru.

[1:13] Stisknutím kláves ‘s’ a ‘enter’ je ukončeno nahrávání, data jsou uložena do souboru a eye tracker je odpojen. Uživatel je všem informován.

[1:20] Ve stejné složce, v jaké je uložen zdrojový kód, je zkontrolován vytvořený soubor a kvalita nahraných dat