Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Eye tracking pomocí kamery The Eye Tribe

Diplomová práce

Autor: Bc. Jan Novotný
Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. Mgr. Tomáš Kozel, Ph.D.

Odborný konzultant: Jan Tomáš
CIRCUS DESIGN s.r.o.

Hradec Králové

duben 2016

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 20. 4. 2016

Poděkování:

Rád bych touto cestou poděkoval doc. Mgr. Tomáši Kozlovi, Ph.D. za cenné rady, ochotu a čas, který mi věnoval při konzultacích.

Rovněž bych chtěl poděkovat Janu Tomášovi a Aleši Rubášovi z UX agentury CIRCUS DESIGN s.r.o. za pomoc, vstřícnost a poskytnutí zázemí při tvorbě práce.

Anotace

První část diplomové práce je obecně zaměřena na problematiku sledování pohybu očí. Jsou popsány principy, na kterých pracují oční kamery, a také zmíněny různé druhy těchto zařízení. Obecně jsou popsány vizualizační techniky, které lze pro analýzu dat ze sledování pohybu očí využít. Detailněji je popsána oční kamera The Eye Tribe. Hlavním cílem práce je analýza, návrh a implementace softwaru k vyhodnocování dat z uživatelského testování s oční kamerou The Eye Tribe. Navržený software je postaven na vícevrstvé architektuře s webovým i desktopovým klientem. Jsou zmíněny některé specifické problémy spojené s vývojem softwaru pro eye tracking. V rámci práce byly navrženy a implementovány algoritmy pro vykreslování vizualizačních technik specifických pro eye tracking.

Annotation

Title: Eye Tracking Using The Eye Tribe Tracker

The first part of this diploma thesis is focused on the eye tracking issue. There are described the principles of eye trackers and also there are mentioned various types of these devices. There are described some eye tracking visualization techniques. There is a focus on The Eye Tribe tracker. The main goal of this diploma thesis is to analyze, design and implement the eye tracking software. This software is using the multitier architecture with a web-based client and a desktop client as well. There are mentioned some specific issues associated with the development of the eye tracking software. In this diploma thesis there were designed and implemented the eye tracking specific visualization algorithms.

Obsah

1 Úvod ... 1
2 Eye tracking ... 2
2.1 Historie ... 2
2.2 Metody měření ... 2
2.2.1 Metoda využívající kontaktní čočky ... 3
2.2.2 Elektrookulografie ... 3
2.2.3 Videookulografie ... 4
2.3 Pohyb lidského oka ... 6
2.4 Oční kamery typu head-mounted a table-mounted ... 7
2.4.1 Head-mounted ... 7
2.4.2 Table-mounted ... 7
2.5 Monokulární a binokulární oční kamery ... 8
2.6 Kalibrace očních kamer ... 8
2.7 Využití sledování pohybu očí ... 9
2.8 Uživatelská testování s oční kamerou ... 9
2.9 Vizualizace naměřených dat ... 10
2.9.1 Heatmap ... 10
2.9.2 Gaze plot ... 12
2.9.3 Gaze video ... 12
2.9.4 Bee swarm ... 13
3 Oční kamera The Eye Tribe ... 14
3.1 Hardwarové parametry ... 15
3.2 Potřebný software ... 15
3.3 Kalibrace oční kamery The Eye Tribe ... 16
3.4 EyeProof ... 16
4 Použité technologie ... 18
4.1 Spring Framework ... 18
4.1.1 Moduly ... 18
4.1.2 Spring Data JPA ... 18
4.1.3 Spring Roo ... 19

4.2 Hibernate 19
4.3 Maven 21
4.4 JavaServer Pages 21
4.4.1 JSTL 22
4.5 JavaFX 22
4.6 FFmpeg a Xuggler 23
4.7 The Eye Tribe Java SDK 23
5 Analýza a návrh softwaru 25
5.1 Diagram případů užití 25
5.2 Diagram tříd 26
5.2.1 Knihovna pro generování vizualizací 28
5.3 Architektura aplikace 30
6 Implementace 33
6.1 Serverová část 33
6.2 Datová část 33
6.3 Klientská část 34
6.3.1 Desktopový klient 34
6.3.2 Webový klient 38
6.4 Zabezpečení 39
6.4.1 Zabezpečení na straně serveru 39
6.4.2 Zabezpečení na straně desktopového klienta 40
6.5 Implementace vizualizačních algoritmů 40
6.5.1 Algoritmus vykreslení heat mapy 40
6.5.2 Algoritmus vykreslení vizualizační techniky gaze plot 44
6.5.3 Rychlost algoritmů 46
7 Výsledky 48
8 Závěr 53
9 Seznam použité literatury 54
10 Přílohy 56

Seznam obrázků

Obr. 1: Kontaktní čočka s cívkou ... 3
Obr. 2: Elektrookulografie ... 4
Obr. 3: Oční kamera umístěná před participanta ... 4
Obr. 4: Oční kamera umístěná na hlavě ... 4
Obr. 5: Pozice zornice a odrazu světla od rohovky při změně směru pohledu ... 5
Obr. 6: Pozice zornice a odrazu světla od rohovky při pohybu hlavy ... 6
Obr. 7: Heat mapa ... 11
Obr. 8: Gaze plot ... 12
Obr. 9: Bee swarm ... 13
Obr. 10: Oční kamera The Eye Tribe ... 14
Obr. 11: EyeTribe UI ... 16
Obr. 12: Diagram případů užití ... 25
Obr. 13: Diagram tříd ... 27
Obr. 14: Diagram tříd - knihovna pro generování vizualizací ... 29
Obr. 15: Architektura aplikace ... 31
Obr. 16: Obrázek, který využívá algoritmus vykreslení heat mapy ... 41
Obr. 17: Pomocný obraz při vytváření heat mapy ... 41
Obr. 18: Stupnice barev od studených po teplé ... 42
Obr. 19: Chyba algoritmu při velkém množství vstupních bodů ... 42
Obr. 20: Heat mapa vygenerovaná navrženým algoritmem ... 43
Obr. 21: Množina vstupních bodů a jejich pořadí ... 44
Obr. 22: Gaze plot vygenerovaný navrženým algoritmem ... 45
Obr. 23: Snímek obrazovky webového klienta – přehled ... 48
Obr. 24: Snímek obrazovky webového klienta – detail studie ... 49
Obr. 25: Export konkrétního měření – heat mapa a gaze plot ... 50
Obr. 26: Snímek videozáznamu z testování ... 51
Obr. 27: Snímek obrazovky desktopového klienta ... 52

Seznam tabulek

Tab. 1: Techniky vizualizace dat 10
Tab. 2: Rychlost algoritmu generování heat mapy 46
Tab. 3: Rychlost algoritmu generování vizualizace gaze plot 47

1 Úvod

Eye tracking neboli sledování pohybu očí je v posledních letech na vzestupu především díky lepší dostupnosti a použitelnosti potřebných technologií. I přesto je však nezbytné hardwarové a softwarové vybavení obecně považováno za velmi finančně nákladné. Na dnešním trhu existuje mnoho očních kamer od různých výrobců. Za nejlevnější oční kameru se prohlašuje produkt od dánské společnosti The Eye Tribe se stejnojmenným názvem. Tato oční kamera je k dostání za 99 amerických dolarů.

V současné době není oční kamera The Eye Tribe dodávána s žádnou aplikací pro běžné použití a je proto určena především pro vývojáře softwaru. To je hlavní motivací pro tvorbu této práce. Ve spolupráci se společností CIRCUS DESIGN s.r.o. byl v rámci práce navržen a implementován software k vyhodnocování dat z uživatelského testování s oční kamerou The Eye Tribe.

Začátek práce je zaměřen na obecnou problematiku sledování pohybu očí. Jsou zmíněny různé technologie pro eye tracking, popsány typy očních kamer a v neposlední řadě také charakterizovány možnosti vizualizace naměřených dat. Následuje popis oční kamery The Eye Tribe.

V práci jsou stručně popsány softwarové technologie použité při tvorbě výsledné aplikace. Další část práce je věnována analýze, návrhu a implementaci aplikace včetně popisu navržených algoritmů pro vizualizaci naměřených dat z oční kamery.

2 Eye tracking

Eye tracking (sledování pohybu očí) je proces, při kterém slouží poloha oka k určení směru pohledu dané osoby. [17]

2.1 Historie

V průběhu let byly vyvinuty různé techniky pro sledování pohybu očí. Počátky výzkumu sledování pohybu očí sahají do konce 19. století. V roce 1879 popsal francouzský oční lékař Louis Emile Javal pohyb lidských očí při čtení textu. Pozoroval, že pohyb oka není plynulý, ale je složený z rychlých pohybů (sakády) a krátkých zastávek (fixace). [17]

Později sestrojil Edmund Huey zařízení, které bylo schopné sledovat pohyb očí při čtení textu. Toto zařízení vyžadovalo, aby měla zkoumaná osoba nasazené čočky s otvorem pro zornici. K čočkám byl připojen hliníkový ukazatel, z jehož pohybu se dalo vyvodit, kam se osoba dívá. [17]

Dále byly vymyšleny i přístupy, které spoléhaly na připnutí elektrod na hlavu v blízkosti očí. Obě tyto možnosti byly však pro zkoumanou osobu nepohodlné a rušivé. [6]

První nerušivá zařízení vznikala na počátku 20. století. Jedno z prvních zhotovil Charles H. Judd. To nahrávalo pohyb oka a pořízený záznam bylo později možné analyzovat. [16]

Eye tracking byl v té době používán především akademiky a výzkumnými pracovníky v medicíně. Rozmach této oblasti nastal především na konci 20. a začátkem 21. století díky lepší dostupnosti a použitelnosti potřebných technologií. [6]

2.2 Metody měření

Zařízení pro sledování pohybu očí jsou založena na několika různých principech. Ty lze dle [17] rozdělit do 3 kategorií:

1) Metoda využívající kontaktní čočky
2) Elektrookulografie
3) Videookulografie

2.2.1 Metoda využívající kontaktní čočky

První z metod sledování pohybu očí spočívá v užití speciálních kontaktních čoček. K těmto čočkám může být připojeno např. malé zrcadlo a podle odrazu světla od zrcadla se dá vyvodit směr pohledu. Dalším řešením je připojení cívky ke kontaktní čočce. K rozeznání pohybu oka se pak využívá princip indukce elektrického proudu. Takovouto speciální kontaktní čočku zobrazuje Obr. 1.

Obr. 1: Kontaktní čočka s cívkou

Zdroj: převzato z [17]

I přes přesnost systémů využívajících kontaktní čočky nejsou tyto metody příliš rozšířené. Hlavním důvodem je nutnost nasazování kontaktních čoček, a tedy značná nekomfortnost. [17]

2.2.2 Elektrookulografie

Elektrookulografie byla nejvíce rozšířená metoda sledování pohybu očí především před zhruba 40 až 50 lety, ale je využívána dodnes. [8] Principem metody je měření elektrických potenciálů oka pomocí elektrod umístěných poblíž oka. Z důvodu velké hustoty nervů sítnice se lidské oko chová jako elektrický dipól, rohovka je kladným pólem a zadní část oka záporným. S pohybem oka se pohybuje i jejich elektrické pole a lze tak zjistit pozici oka pomocí elektrod umístěných v jeho blízkosti. [17]

Měření pomocí elektrod připevněných k hlavě zobrazuje Obr. 2.

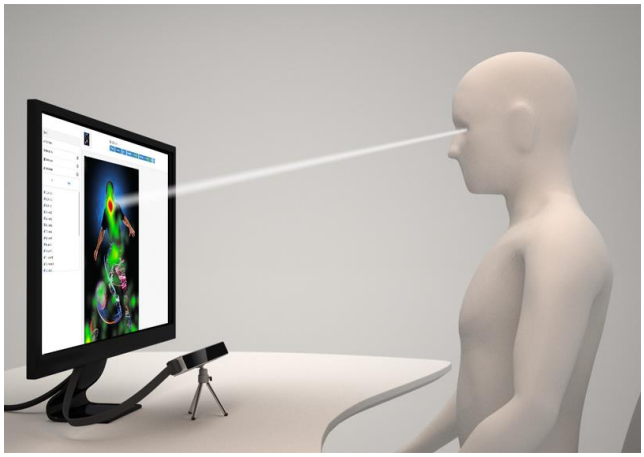
Obr. 2: Elektrookulografie

Zdroj: převzato z [8]

Tuto metoda lze použít i v případě zavřeného oka. Její nevýhodou je nutnost připnutí elektrod na hlavu participanta.

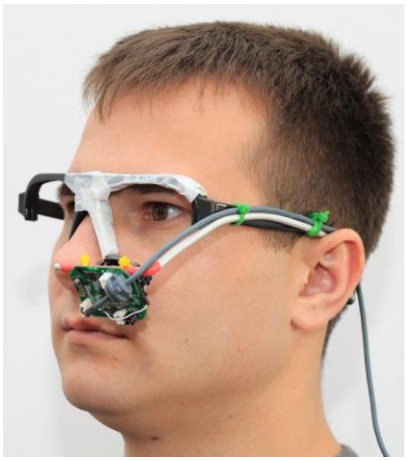
2.2.3 Videookulografie

Videookulografie je skupina metod, které používají ke sledování pohybu oka videozáznam. Výhodou je možnost vzdáleného měření, tudíž může být potřebné zařízení, nazývané také oční kamera (eye tracker), umístěno před participanta (znázorňuje Obr. 3) anebo nějakým způsobem umístěné na hlavě, obvykle ve formě brýlí (Obr. 4). [17]



Obr. 3: Oční kamera umístěná před participanta

Zdroj: převzato z [23]



Obr. 4: Oční kamera umístěná na hlavě

Zdroj: převzato z [17]

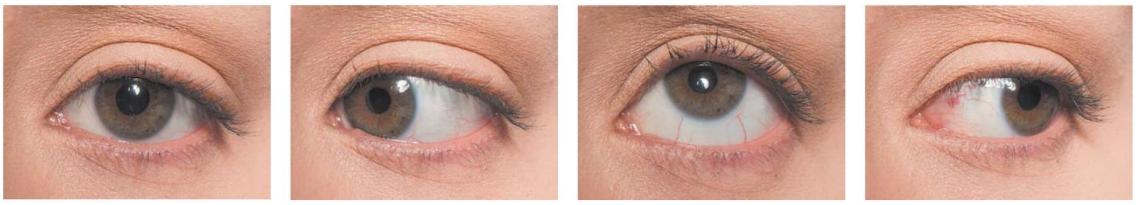
Oční kamery na tomto principu mohou zpracovávat obraz jak z viditelného, tak i infračerveného spektra. Zpracovávání obrazu viditelného spektra spoléhá na odraz světla od oka, avšak při špatných světelných podmínkách je velmi obtížné určit směr pohledu. Řešením tohoto problému je využití infračerveného (lidským okem neviditelného) spektra. [17]

Oční kamery ozařují obličej infračerveným světlem a nahrávají dvě věci:

- odraz tohoto světla od sítnice, který pomáhá nalézt střed zornice,
- odraz od rohovky (corneal reflection).

Směr pohledu je poté možné dopočítat z relativní pozice středu zornice a odrazu od rohovky. Výhodou infračerveného světla je, že je pro lidské oko neviditelné, tudíž neruší a neoslňuje jako viditelné světlo, stále si však zachovává schopnost odrazu od lidského oka. Oční kamery vyzařují infračervené světlo pouze s malou intenzitou, takže není pro lidi nebezpečné. [6]

Relativní pozice středu zornice a odrazu světla (v tomto případě viditelného) od rohovky při změně směru pohledu znázorňuje Obr. 5.



Obr. 5: Pozice zornice a odrazu světla od rohovky při změně směru pohledu

Zdroj: převzato z [6]

Výše zmíněné metody sledování pohybu očí (metoda využívající kontaktní čočky a elektrookulografie) jsou sice vhodné pro měření pohybu oka, k určení směru pohledu však požadují, aby byla hlava participanta zafixovaná na stále stejném místě a nehýbala se. Videookulografie však dokáže počítat i s pohybem hlavy. [8]

Jak znázorňuje Obr. 6, relativní pozice zornice a odrazu světla od rohovky se při pohybu hlavy nemění, jestliže se participant dívá stále na stejné místo.

Obr. 6: Pozice zornice a odrazu světla od rohovky při pohybu hlavy

Zdroj: převzato z [6]

I přesto, že každá ze zmíněných metod měření má své pro a proti, pravděpodobně nejpraktičtější je využití metody založené na odrazu světla od rohovky oka (videookulografie). Výstupem na tomto principu pracujících očních kamer mohou být souřadnice bodů (x, y) na monitoru, na které se participant dívá. Výhodou těchto očních kamer je, že participanta při měření neruší a také jejich dostatečná přesnost. Omezení těchto očních kamer je v jejich vzorkovací frekvenci. Ta je typicky limitována snímkovou frekvencí videa – obvykle 60 Hz. Při frekvenci 60 Hz je z tohoto důvodu nutné očekávat obdržení dat z kamery každých cca 16 až 17 milisekund. [8]

2.3 Pohyb lidského oka

Pohyb lidského oka se dá rozdělit na několik druhů, z nichž nejdůležitější pro eye tracking jsou sakády a fixace. Pohled lidí „skáče“ z místa na místo průměrně třikrát až čtyřikrát za vteřinu. [6] Tyto rychlé pohyby jsou nazývány sakády a zrakové vnímání je při nich omezeno. Ostře vidí lidé pouze během fixace, kdy jsou oči relativně nehybné a soustředí se na jedno místo. Přesun fixačních bodů z jednoho místa na druhé, které si mozek předem vyhlédl pomocí periferního vidění, je uskutečňován pomocí sakád. [15] Délka fixace bývá obvykle mezi desetinou až polovinou sekundy [6], sakády jsou kratší – obyčejně zhruba 0,05 vteřiny. [15]

2.4 Oční kamery typu head-mounted a table-mounted

Jak uvádí zdroje [6] a [8], oční kamery se dají rozdělit na ty, které se umisťují před participanta (table-mounted) a na ty, které se umisťují na hlavu (head-mounted).

2.4.1 Head-mounted

Head-mounted systémy (v knize [6] jsou také nazývány „*wearable eye trackers*“) jsou připevněny na hlavu např. pomocí brýlí, čelenky či pokrývky hlavy (Obr. 4). Jsou využívány především ve studiích, ve kterých je zapotřebí, aby se participant pohyboval a interagoval s fyzickými objekty. Příkladem využití může být získávání dat při nakupování v obchodě, v terénu a podobně. [10]

Výhodou těchto zařízení je možnost volnosti v pohybu participanta. Nicméně většina těchto kamer poskytuje nejlepší výsledky v případě, že je zkoumaný objekt ve stejné vzdálenosti, pro jakou byla provedena kalibrace. [6] Pokud je objekt dále či blíže, nemusí být výsledky přesné (anglicky se tato chyba nazývá parallax error a je detailněji popsána např. v článku [10]).

Nevýhodou těchto očních kamer je jejich rušivost pro participanta, jelikož musí být připevněné na hlavě a obvykle je alespoň jejich část při měření v zorném poli participanta. Další nevýhodou je složitější analýza získaných dat, která je většinou manuální. To je způsobeno tím, že se scéna v průběhu měření mění. [6]

2.4.2 Table-mounted

Table-mounted systémy (v knize [6] nazývány také jako „*remote eye trackers*“) se umisťují při měření před participanta, zpravidla na stůl pod monitor (Obr. 3). [8] Typicky jsou využívány ve studiích, při kterých participant sedí či stojí na jednom místě a zkoumaný kontext (např. webové stránky) mu je zobrazován na monitoru. [6]

Výhodou oproti head-mounted očním kamerám je, že participanta nijak neobtěžují. Ten tak může jednoduše zapomenout na to, že je nahráváno, kam se dívá. V případě, že je zkoumán statický kontext (např. obrázek), bývá jednodušší analýza dat, jelikož zobrazovaný kontext se pro různá měření neliší. [6]

Nevýhodou je omezení pohybu při měření. Participant musí sedět naproti oční kameře a pro správnost měření jsou přijatelné pouze menší pohyby hlavou. [6]

2.5 Monokulární a binokulární oční kamery

Oční kamery lze rozdělit na monokulární a binokulární. Pro určení směru pohledu nahrávají monokulární oční kamery pouze jedno oko, binokulární obě. Pro účely sledování pohybu očí stačí teoreticky využít pouze jedno oko, poněvadž se obě pohybují současně (pokud participant netrpí nějakou oční vadou). V praxi je ale lépe využít binokulární systémy, které mají vyšší přesnost – především díky tomu, že lze průměrovat získaná data z obou očí. Další výhodou je, že v případě dočasné ztráty dat získaných ze sledování jednoho oka (což může být způsobeno např. pohybem hlavy mimo sledovanou oblast oční kamery) jsou stále k dispozici data ze sledování druhého oka. [6]

2.6 Kalibrace očních kamer

Jak uvádějí knihy [8] a [6], většina dnešních očních kamer vyžaduje pro svou správnou funkčnost kalibraci. Ta je prováděna na začátku každého sezení před začátkem sběru dat. Při procesu kalibrace je po participantovi požadováno, aby se díval na sérii bodů, které mu jsou postupně zobrazovány (typicky na monitoru). Tyto body jsou většinou zobrazeny ve formě puntíků anebo křížků. Kalibraci lze provést také pomocí bodů znázorněných na tabuli či papíře. V případě, že je měření s oční kamerou prováděno ve venkovním prostředí (čímž může být použití během řízení auta či při chůzi), používají se různé body umístěné v daném prostředí. [6]

Principem kalibrace je zobrazení posloupnosti viditelných bodů v mezních rozsazích pozorovacích úhlů, typicky např. levý horní, pravý horní, levý dolní a pravý dolní roh monitoru. Tyto informace jsou poté interpolovány či extrapolovány při výpočtu dalších možných směrů pohledu. [6]

Většina očních kamer poskytuje vlastní vestavěné kalibrační techniky. Typicky je při kalibraci zobrazováno participantovi 3, 5 anebo 9 bodů. [8] Pro dosaženích lepších výsledků kalibrace je vhodné jich využít až 12 či 16. [23]

2.7 Využití sledování pohybu očí

Způsob využití očních kamer se dá rozdělit do dvou kategorií: aktivní a pasivní. [23] Zdroj [6] nazývá aktivní přístup také jako „*input device*“ a pasivní jako „*research technique*“.

Aktivní využití umožňuje uživateli oční kamery zužitkovat pohyby oka k ovládání různých zařízení, aplikací, počítačových her a tak podobně. Pro přirozenější a komfortnější ovládání je vhodné nevyužívat pouze pohyb očí, ale kombinovat ho s dalšími periferiemi jako klávesnice, myš, dotyková obrazovka či různá tlačítka. Mezi příklady takového využití patří např.: [23]

- automatické posouvání webové stránky či různých dokumentů při pohledu na spodní/horní část dokumentu,
- označení ikony v operačním systému, pokud se na ni uživatel podívá,
- zobrazení klávesnice na monitoru, která je určena pro psaní textu uživatelům s handicapem, či jiný způsob přizpůsobení ovládání počítače handicapovaným uživatelům
- a mnoho dalších.

Při pasivním využití jsou sbírána a vyhodnocována objektivní data o tom, kam se uživatel dívá. Oční kamera může být využita ke sběru dat, jsou-li participantovi zobrazovány např. webové stránky, uživatelská rozhraní softwaru, různý reklamní materiál a podobně. [23]

2.8 Uživatelská testování s oční kamerou

Uživatelské testování použitelnosti je výzkumná technika, která hodnotí snadnost použití produktu (např. webová stránka či software). Při uživatelském testování je požádán reprezentativní vzorek uživatelů, aby produkt reálně používal. Výzkumníci poté pozorují a zaznamenávají, co participanti říkají a dělají. Na základě získaných údajů pak lze vydat doporučení o různých vylepšeních. Při uživatelském testování lze využít i oční kameru. Ta dodává výzkumníkům další informace – o tom, kam se participant při testování díval. Kombinace sledování

pohybu očí s uživatelským testováním tak může pomoci vysvětlit rozhodování participantů anebo zjistit, co považují za zajímavé, co zaujme jejich pozornost. [5]

2.9 Vizualizace naměřených dat

Dle [6] se dá vizualizace naměřených dat z oční kamery rozdělit dle:

- množství zobrazovaných dat – zobrazení dat z měření jednoho participanta anebo agregace dat od více participantů,
- formátu – vizualizace statická či dynamická,
- typu zobrazených informací – vizualizace pouze prostorových dat (kam se participant díval) či vizualizace prostorových a časových dat (kam se díval v jakém čase).

Softwary vyhodnocující eye tracking běžně používají různé techniky vizualizace naměřených dat. Názvy technik a jejich rozdělení zobrazuje Tab. 1.

Tab. 1: Techniky vizualizace dat

Název	Množství zobrazovaných dat	Formát	Typ zobrazených informací
Heatmap	Více participantů	Statický	Prostorové
Gaze plot	1 participant	Statický	Prostorové i časové
Gaze video	1 participant	Dynamický	Prostorové i časové
Bee Swarm	1 i více participantů	Dynamický	Prostorové i časové
Dynamic Heatmap	Více participantů	Dynamický	Prostorové i časové

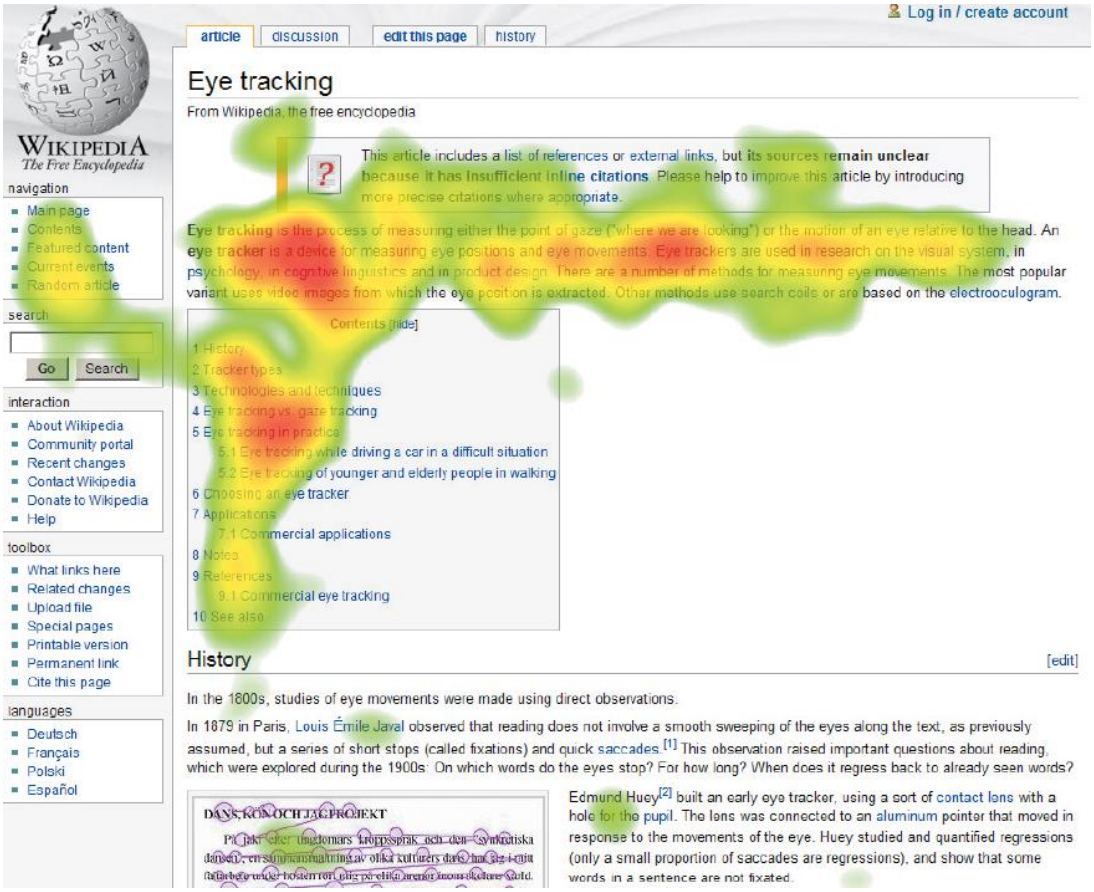
Zdroj: přeloženo a upraveno z [6]

2.9.1 Heatmap

Dle zdrojů [4] a [6] je heatmap (česky teplotní mapa anebo také počeštěně heat mapa) dvourozměrná grafická reprezentace dat, která znázorňuje, kam se participant při měření díval/nedíval. V heat mapách jsou barevně vyobrazeny oblasti zájmu, tedy místa, na která směřoval pohled. Barvy jsou obvykle brány z přirozené stupnice barev od studených po teplé. Oblasti, kam se uživatel díval

nejméně, jsou běžně znázorněny modře či zeleně, dále následuje žlutá a oranžová barva, nejpozorovanější části jsou pak vyobrazeny červeně.

Typickou heat mapu zobrazuje Obr. 7.



Obr. 7: Heat mapa

Zdroj: převzato z [6]

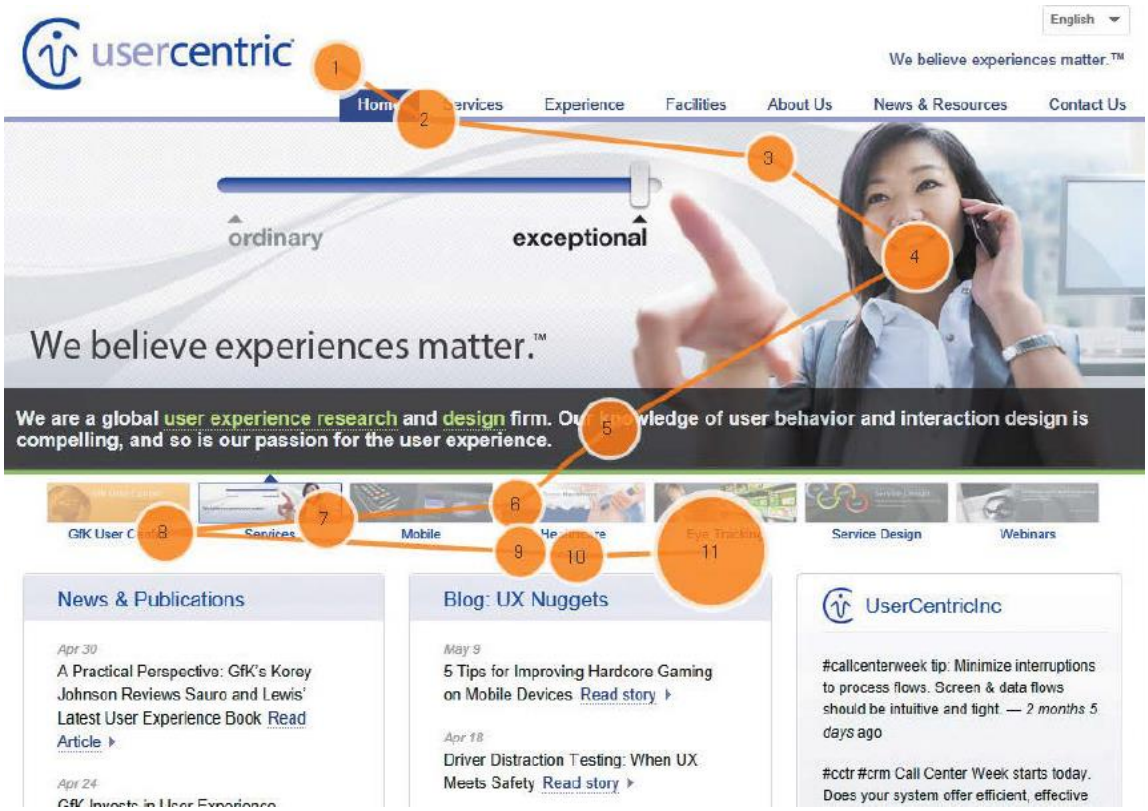
Ačkoli mohou být tvořeny heat mapy z dat pouze jednoho měření, mnohem užitečnější jsou při vyobrazení agregovaných dat od více participantů. [6]

Podobná technika teplotním mapám se nazývá gaze opacity map [4] nebo také focus map [6]. Jedná se o techniku vizualizace založenou na stejném principu. Namísto využití stupnice barev od studených po teplé je ale použita škála průhlednosti. Stimul (obrázek, webová stránka apod.) je překryt obvykle černou barvou a místa, kam se uživatel díval nejvíce, jsou průhledná. Výsledem je tedy vyobrazení oblastí zájmu, ostatní oblasti jsou zakryty neprůhlednou barvou. [6]

2.9.2 Gaze plot

Jak uvádí zdroje [4] a [6], technika vizualizace nazvaná gaze plot vyobrazuje cestu, kterou prošel pohled participanta při měření. Každá fixace pohledu je znázorněna jako kruh s číslem a každá sakáda jako čára. Velikost kruhu závisí na délce fixace – čím delší je fixace, tím větší je kruh. Čísla v kruhu znázorňují pořadí fixací v průběhu času – k první fixaci je připojeno číslo 1, k druhé 2 atd.

Příklad vizualizační techniky gaze plot zobrazuje Obr. 8.



Obr. 8: Gaze plot

Zdroj: převzato z [6]

2.9.3 Gaze video

Výše zmíněné techniky (Heat mapa a gaze plot) vyžadují, aby byl participantovi zobrazován statický kontext, jako jsou např. obrázky. To může být v některých případech limitující – např. při sledování pohybů očí u videa či webových stránek, jejichž podoba se může v průběhu měření měnit. Pro takovéto případy užití je vhodné využít tzv. gaze video. Jedná se o dynamickou vizualizaci, kterou lze

přehrát stejně jako klasické video. Na snímcích je vždy stejný podklad, jaký viděl reálně participant, a v popředí je vykreslen kruh či křížek v místě, kam směřoval jeho pohled. [6]

2.9.4 Bee swarm

Pokud je zaručeno, že byl všem participantům při měření zobrazován ve stejný čas totožný kontext (např. obrázek či video), je možné využít tzv. bee swarm. Jedná se taktéž o dynamickou vizualizaci, kterou lze přehrát jako klasické video. Rozdílem oproti technice gaze video je v tom, že je možné využít data od všech participantů, ne pouze od jednoho. Na konkrétních snímcích je pak vždy vykreslen podklad, který v daný okamžik všichni viděli a do popředí jsou vyobrazeny kruhy či křížky (obvykle různě zbarvené) v oblastech, kam směřovaly jednotlivé pohledy. [6]

Tato vizualizační technika nelze využít např. při testování webových stránek, jelikož každý uživatel posouvá obsah webu či kliká na odkazy v jiný časový okamžik. [6]

Namísto kruhů či křížků může být postupně vykreslována také heat mapa. Tato technika se nazývá dynamic heatmap. [6]

Snímek z techniky bee swarm zobrazuje Obr. 9.

Obr. 9: Bee swarm

Zdroj: převzato z [6]

3 Oční kamera The Eye Tribe

Na dnešním trhu existuje mnoho očních kamer od různých společností. Obecně jsou tyto technologie považovány za velmi drahé. Za nejlevnější oční kameru se prohlašuje produkt od dánské společnosti The Eye Tribe se stejnojmenným názvem. Tato oční kamera je k dostání za 99 amerických dolarů. V lednu roku 2016 byla oznámena další verze oční kamery nazvaná The Eye Tribe Tracker PRO za 199 dolarů, jejíž očekávaný termín vydání je červen 2016. [18]

The Eye Tribe je binokulární oční kamera pracující na principu videookulografie. Pro rozpoznání směru pohledu využívá infračerveného světla. Jedná se o oční kameru umisťovanou před participanta (table-mounted). Součástí kamery je i odnímatelný stojánek pro snadné umístění na stůl, nejlépe pod monitor počítače. [18]

V době psaní práce (leden 2016) není oční kamera The Eye Tribe dodávána a svázána s žádnou aplikací pro běžné použití, a je proto určena především pro vývojáře softwaru. [18]

Zařízení The Eye Tribe zobrazuje Obr. 10.

Obr. 10: Oční kamera The Eye Tribe

Zdroj: vlastní

3.1 Hardwarové parametry

Vzorkovací frekvence oční kamery The Eye Tribe je defaultně 30 Hz, lze však nastavit i 60 Hz. Ideální vzdálenost očí od kamery by měla být v rozmezí 45 – 65 centimetrů. Maximální udávaná podporovaná velikost monitoru je 24 palců. Zařízení má tvar kvádru s rozměry 20 × 1,9 × 1,6 cm a hmotnost 70 gramů. Oční kamera je optimalizována pro zapojení pomocí USB 3.0, nefunguje s USB 2.0. Kamera by si měla také poradit s většinou brýlí a ani kontaktní čočky by jí neměly dělat žádné problémy. [18]

Přesnost určení směru pohledu je udávána s přesností 0,5° až 1°. Za předpokladu, že je uživatel vzdálen od monitoru (resp. oční kamery) cca 60 cm, odpovídá této přesnosti odchylka 0,5 až 1 cm na monitoru. [23]

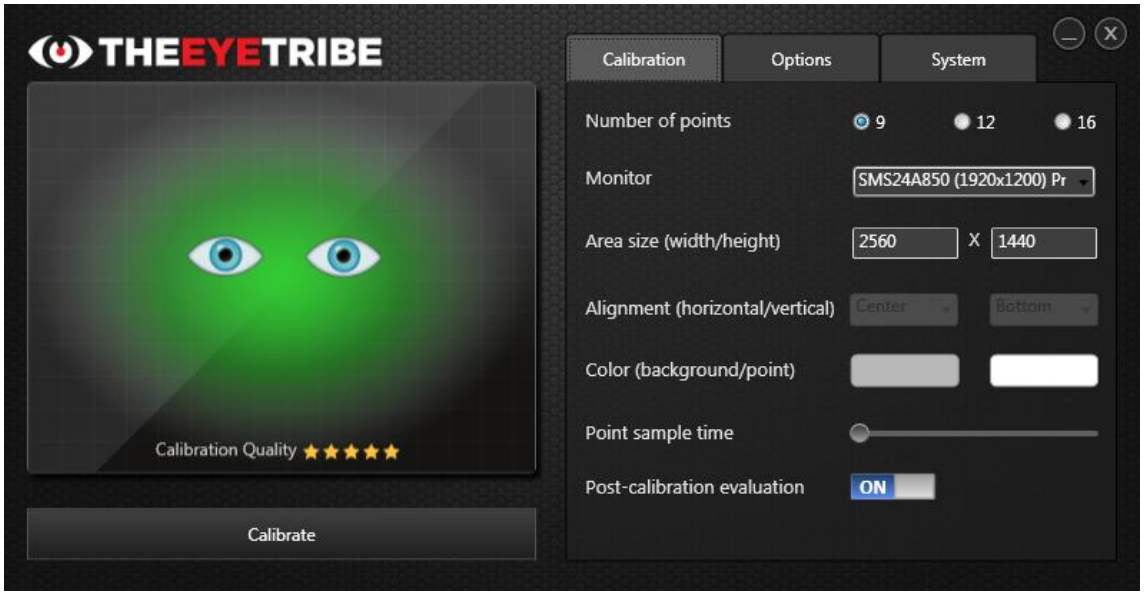
3.2 Potřebný software

Pro správnou funkčnost oční kamery The Eye Tribe je nutná instalace softwaru, který je ke stažení na stránkách výrobce [23] (je nutné se přihlásit k účtu, pomocí kterého byla kamera objednána a koupena). Jedná se o sadu dvou aplikací: EyeTribe Server a EyeTribe UI.

Komunikace aplikací s oční kamerou je založena na otevřeném API (Application Programming Interface). Po připojení oční kamery je pro její používání nutný běh aplikace EyeTribe Server. Ta funguje jako prostředník mezi hardwarem a uživatelským softwarem. Komunikace konkrétních aplikací s oční kamerou tak probíhá skrze EyeTribe Server pomocí TCP (Transmission Control Protocol) soketů. Zprávy jsou ve formátu JSON (JavaScript Object Notation). Díky tomuto principu komunikace lze pro tvorbu vlastního softwaru využít jakýkoli programovací jazyk, který dokáže pracovat s TCP sokety. Pro snadnější vývoj však výrobce již předpřipravil SDK (software development kit) pro jazyky C#, C++ a Java. Jedná se o knihovny s dostupnými zdrojovými kódy, tudíž si je může každý upravit dle libosti. [23]

3.3 Kalibrace oční kamery The Eye Tribe

Hlavním účelem aplikace EyeTribe UI je kalibrace oční kamery Eye Tribe. Stejně jako všechny ostatní aplikace komunikuje s kamerou skrze EyeTribe Server. Grafické rozhraní aplikace zobrazuje Obr. 11.



Obr. 11: EyeTribe UI

Zdroj: převzato z [23]

Po spuštění aplikace EyeTribe UI je uživateli zobrazen tzv. trackbox, který zobrazuje v reálném čase informace o relativní poloze jeho očí vůči oční kameře. Pomáhá tak nalézt optimální polohu hlavy vůči kameře pro co možná nejpřesnější výsledky. [23]

Proces kalibrace oční kamery Eye Tribe pomocí EyeTribe UI trvá obvykle přibližně 20 vteřin. Participant se při něm musí dívat na sekvenci postupně objevujících se bodů na monitoru. Počet bodů lze předem nastavit na 9, 12 či 16. Po kalibraci je uživateli zobrazen tzv. evaluation view, který pomáhá ověřit, zda se kalibrace zdařila dostatečně dobře. [23]

3.4 EyeProof

Ačkoli není oční kamera The Eye Tribe dodávána s žádným softwarem pro běžné využití, vyvíjejí vývojáři zařízení cloudový projekt nazvaný EyeProof. V současné

době je projekt zatím pouze v uzavřené beta verzi – po registraci na webu projektu [11] je nutné vyčkat na přijetí. Jedná se o analytický software pro eye tracking, pomocí něhož lze testovat obrázky a webové stránky. Pro testování je nutné mít v počítači nainstalováno klienta, který po měření odešle data z oční kamery na server, kde je později možné tato data analyzovat. Pro účely analýzy používá EyeProof vizualizační techniky gaze plot, bee swarm a heatmap (popř. gaze opacity). Pro zobrazení heat mapy či bee swarm mohou být použita i agregovaná data od všech participantů. [11]

Nevýhodou softwaru EyeProof je, že dokáže pracovat pouze se statickým kontextem (obrázky). V případě testování webových stránek je nejprve web vyrenderován jako obrázek, který je pak participantovi zobrazen na monitoru. Není tak možné sledovat chování participanta při procházení webu.

4 Použité technologie

V této části práce jsou popsány technologie, které byly využity při tvorbě aplikace k vyhodnocování dat z uživatelského testování s oční kamerou. Je popsána také knihovna „*The Eye Tribe Java SDK*“, která usnadňuje vývoj softwaru komunikujícího s oční kamerou The Eye Tribe.

Pro vývoj aplikace byla využita platforma Java především pro její rozšířenost, podporu oční kamery The Eye Tribe a předchozí zkušenosti autora.

4.1 Spring Framework

Spring Framework je komplexní open-source aplikační rámec, který poskytuje obsáhlou infrastrukturu pro vývoj aplikací na platformě Java. Poskytuje možnost sestavení aplikace z obyčejných Java tříd (Plain Old Java Objects – POJO). Vývojář se díky němu může soustředit na vývoj svého specifického softwaru. [20]

4.1.1 Moduly

Spring Framework je navržen jako modulární systém. Funkčnost celého frameworku je rozdělena do několika modulů. Vývojář softwaru tak může použít pouze ty části, které potřebuje. [20]

Spring poskytuje moduly pro podporu různých funkcionalit – dependency injection, expression language, aspektově orientované programování, moduly pro podporu transakcí, testování komponent, zjednodušení práce s databází a v neposlední řadě také modul Web pro usnadnění tvorby webových aplikací. Součástí modulu Web je i nástroj Spring MVC pro tvorbu webových aplikací založených na architektuře MVC (Model-View-Controller). [20]

4.1.2 Spring Data JPA

Součástí modulu Spring Data je tzv. Spring Data JPA. Tento nástroj zjednodušuje tvorbu vrstvy přístupu k datům (data access layer). Cílem je snížení množství často používaného kódu, který je nutný k implementaci této vrstvy.

Příkladem využití Spring Data JPA může být následující kód převzatý z [19], který demonstruje, jak je možné vyhledat uživatele v databázi na základě emailové adresy:

```
public interface UserRepository extends JpaRepository<User, Long> {
  @Query("select u from User u where u.emailAddress = ?1")
  User findByEmailAddress(String emailAddress);
}
```

4.1.3 Spring Roo

Spring Roo je snadno použitelný nástroj pro rychlou tvorbu aplikací s využitím frameworku Spring. Funguje na principu generování zdrojového kódu při vývoji aplikace. Dokáže vygenerovat strukturu projektu s nezbytnými soubory, vygenerovat Java třídy s potřebnými atributy a metodami, vytvořit struktury pro webovou aplikaci s využitím Spring MVC, vygenerovat konfigurační soubory pro přístup k databázi a tak podobně. Využívá principů aspektově orientovaného programování. Defaultně Spring Roo pro práci s datovými zdroji využívá návrhového vzoru Active Record, nicméně lze nastavit i jiný přístup. [21]

4.2 Hibernate

Hibernate je nástroj pro objektově relační mapování (ORM), což je technika, která automaticky obsluhuje persistenci objektů v aplikaci do tabulek v relační databázi. Při využívání ORM frameworků je programátor odstíněn od psaní SQL (Structured Query Language) dotazů, používá techniky daného frameworku. Hibernate je jednou z implementací specifikace JPA (Java Persistence API). [2]

Mapování objektů na databázové tabulky je možné realizovat pomocí XML zápisu anebo anotacemi. [2] Pro třídu *Employee* s dvěma atributy – *id* a *name*, z nichž *id* je databází automaticky generované celé číslo, může jednoduché mapování vypadat např. takto:

```
<class name="Employee" table="employee" catalog="nazevDatabaze">
	<id name="id" type="java.lang.Integer">
		<column name="id" />
		<generator class="identity" />
	</id>
	<property name="name" type="string">
		<column name="name" length="100" not-null="true" />
	</property>
</class>
```

Mapování té samé třídy za pomoci anotací ze specifikace JPA lze realizovat tímto způsobem:

```
@Entity
@Table(name = "employee", catalog = "nazevDatabaze")
public class Employee implements java.io.Serializable {
    private Integer id;
    private String name;

    @Id
    @GeneratedValue(strategy = IDENTITY)
    @Column(name = "id", unique = true, nullable = false)
    public Integer getId() {return this.id;}

    @Column(name = "name", nullable = false, length = 100)
    public String getName() {return this.name;}

    public void setId(Integer id) {this.id = id;}
    public void setName(String name) {this.name = name;}
}
```

Hibernate poskytuje funkcionalitu generování Java tříd i s jejich definovaným mapováním z vytvořených databázových tabulek. Je možné generovat jak XML, tak i variantu s anotacemi. Lze také generovat databázové tabulky s jejich strukturou z předem vytvořených Java tříd. Toto generování lze nastavit v konfiguračním souboru Hibernatu vlastností *hibernate.hbm2ddl.auto* různými způsoby: [2]

- *validate* (nedělá žádné změny v databázi, pouze kontroluje její strukturu),
- *update* (kontroluje a může měnit strukturu databáze),
- *create* (vytváří novou strukturu databáze, maže předchozí i s daty),
- *create-drop* (stejné chování jako *create*, ale navíc na konci sezení maže celou strukturu databáze).

Hibernate poskytuje funkcionalitu jak pro základní CRUD (create, read, update, delete) operace, tak i pro složitější úkony. K tomu lze využít jazyk HQL (Hibernate Query Language), případně JPQL (Java Persistence Query Language). Oba tyto jazyky jsou podobné SQL. Nepracují však s databázovými tabulkami, ale s objekty. [12]

4.3 Maven

Maven je nástroj sloužící pro sestavování a správu projektů na platformě Java. K popisu projektu používá tzv. project object model (POM), ve kterém jsou popsány všechny potřebné informace. Jde o dokument ve formátu XML (Extensible Markup Language) pojmenovaný pom.xml a umístěný v kořenovém adresáři projektu. [7]

Jednou z klíčových funkcí nástroje Maven je správa závislostí projektu. Do souboru pom.xml se definují závislosti, které daná aplikace očekává. Existují dva typy repozitářů – lokální a vzdálené. Maven hledá potřebné knihovny nejprve v lokálním repozitáři, pokud se zde však daná knihovna nenachází, prohledá všechny vzdálené repozitáře, ke kterým má přístup, stáhne a uloží závislosti do lokálního repozitáře. Knihovny z lokálního repozitáře jsou pak použity při sestavování aplikace. [7]

4.4 JavaServer Pages

JavaServer Pages (JSP) je technologie používaná pro tvorbu dynamických webových aplikací na platformě Java. Pro běh JSP je vyžadován webový server s podporou servletového kontejneru, např. Apache Tomcat. [3]

Vložení dynamických prvků do statické HTML (HyperText Markup Language) stránky se provádí pomocí speciálních značek, které se vkládají přímo do HTML stránky. [3]

Jednoduchá ukázka může vypadat např. takto:

```
<h1>Od jedné do sta</h1>
<% for (int i=1; i<=100; i++) { %>
    <%= i %>
<% } %>
```

4.4.1 JSTL

Ačkoli JSP nabízí několik svých značek, tyto značky neposkytují mnoho funkcionality. Z tohoto důvodu vznikla knihovna JSTL (JSP Standard Tag Library), která jich nabízí více. JSTL tak umožňuje nahradit Java kód značkami značkovacího jazyka. [3]

Ukázka využití JSTL:

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<h1> Od jedné do sta </h1>
<c:forEach begin="1" end="100" var="i">
<c:out value="${i}" />
</c:forEach>
```

4.5 JavaFX

JavaFX je nástroj pro tvorbu uživatelských rozhraní klientských aplikací postavených na platformě Java. Je dostupný jako součást JDK 8 (Java Development Kit). JavaFX podporuje vyobrazení 2D a 3D grafiky, různých ovládacích prvků, grafů, přehrávání audia a videa atd. Technologie JavaFX je vhodná jak pro tvorbu desktopových aplikací, tak i za účelem vývoje pro mobilní zařízení a také webových appletů. [9]

Vývoj aplikací v JavaFX lze realizovat dvěma způsoby:

1) Tvorba celého grafického rozhraní v Java kódu.
2) Využití FXML.

FXML je jazyk založený na XML, ve kterém se definuje struktura a rozložení výsledného grafického rozhraní aplikace. K vytváření FXML dokumentů lze využít WYSIWYG (What you see is what you get) editor nazvaný SceneBuilder. [9]

Vzhled JavaFX aplikací lze upravovat pomocí kaskádových stylů. Nejedná se však o standardní CSS (Cascading Style Sheets) navržené konsorciem W3C (World Wide Web Consortium), ze kterého však vychází. Hlavním rozdílem je v názvech CSS vlastností – v JavaFX verzi každá vlastnost začíná prefixem „*-fx-*“. [9]

4.6 FFmpeg a Xuggler

FFmpeg je multiplatformní nástroj pro nahrávání, konverzi a streamování videa a audia. Důležitou součástí je utilita ffmpeg, která se ovládá pomocí příkazové řádky a umožňuje konverzi audia a videa. [24]

Xuggler je open-source Java knihovna pro práci s audiem a videem, která je založena na projektu FFmpeg. [24] Xuggler lze využít např. pro nahrávání dění na obrazovce. Princip tohoto nahrávání dokumentuje následující kód:

```
IMediaWriter writer = ToolFactory.makeWriter("C:/video.mp4");
writer.addVideoStream(0, 0, 1920, 1080);

while (notFinish) {
	BufferedImage screenShot = getDesktopScreenshot();
	writer.encodeVideo(0, screenShot, System.nanoTime() - startTime,
		TimeUnit.NANOSECONDS);
	Thread.sleep(100);
}

writer.close();
```

Výsledkem běhu tohoto kódu je pak soubor „C:/video.mp4“ s nahraným videozáznamem dění na obrazovce.

4.7 The Eye Tribe Java SDK

Jak již bylo zmíněno výše (3.2), výrobce oční kamery The Eye Tribe předpřipravil za účelem snadnějšího vývoje knihovny pro práci s kamerou v jazycích C#, C++ a Java. V práci bude stručně popsána verze pro jazyk Java – nazývaná výrobcem jako „*The Eye Tribe Java SDK*“. [14] Zdrojový kód této knihovny je dostupný na serveru GitHub [22].

Nejdůležitější funkčností této knihovny je bezesporu zjištění souřadnic monitoru, na které se uživatel dívá. Toho lze dosáhnout nejjednodušeji takto:

```
public class SimpleEyeTribe {
    public static void main(String[] args) {
        final GazeManager gm = GazeManager.getInstance();
        boolean success = gm.activate();
        final GazeListener gazeListener = new GazeListener();
        gm.addGazeListener(gazeListener);
    }
    private static class GazeListener implements IGazeListener {
        @Override
        public void onGazeUpdate(GazeData gazeData) {
            System.out.println(
                   "souradnice x: " + gazeData.rawCoordinates.x);
            System.out.println(
                   "souradnice y: " + gazeData.rawCoordinates.y);
        }
    }
}
```

Namísto `rawCoordinates` lze využít i hodnotu `smoothedCoordinates`, která by měla být očištěna od různých rušení.

Zjištění, zda je oční kamera připojena a schopná provozu, lze učinit voláním metody `getTrackerState` třídy `GazeManager`. Ta může vracet několik hodnot, např.: [14]

- `TRACKER_CONNECTED` – oční kamera je správně připojena,
- `TRACKER_NOT_CONNECTED` – oční kamera není připojena,
- `TRACKER_CONNECTED_NOUSB3` – oční kamera je připojena pomocí USB 2.

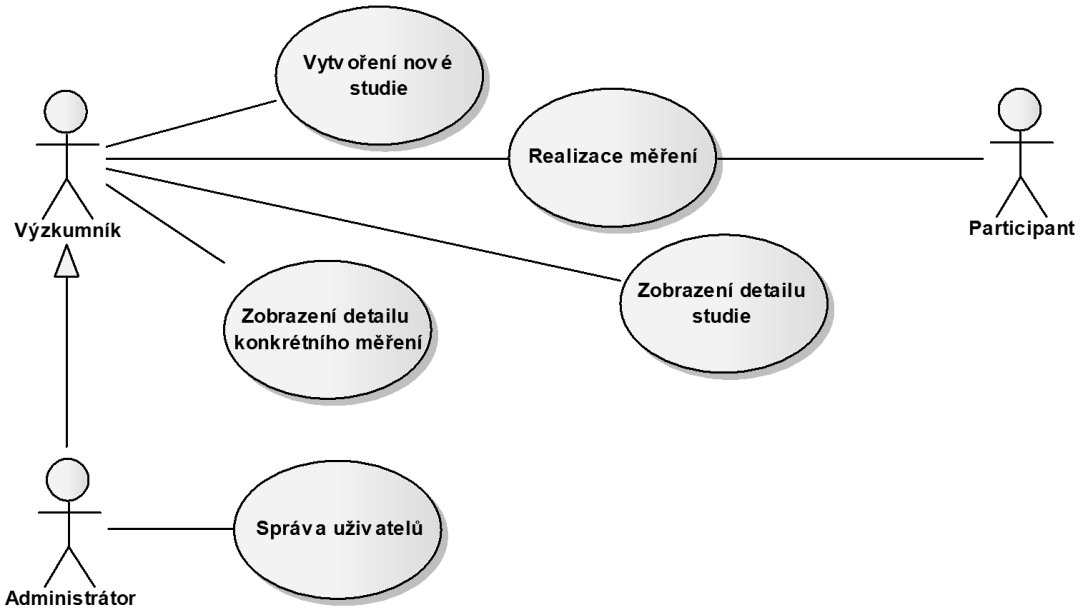
Pro rozpoznání, kdy byla oční kamera připojena či odpojena, je možné využít i rozhraní `ITrackerStateListener`. Pokud třída implementuje toto rozhraní a je zaregistrována pomocí metody `addGazeListener` třídy `GazeManager`, je automaticky zavolána její metoda `onTrackerStateChanged` při každém připojení či odpojení zařízení.

5 Analýza a návrh softwaru

Hlavním cílem práce je navrhnout a implementovat software k vyhodnocování dat z uživatelského testování s oční kamerou The Eye Tribe. V této části práce bude popsána analýza a návrh tohoto softwaru.

5.1 Diagram případů užití

Software byl tvořen především pro účely UX (user experience) agentury CIRCUS DESIGN s.r.o. Po několika konzultacích bylo definováno, k jakým účelům by měl software sloužit a byly tak vymezeny různé požadavky. Z požadavků byl vytvořen diagram případů užití (use case diagram), který zobrazuje hlavní funkcionalitu aplikace. Viz Obr. 12.



Obr. 12: Diagram případů užití

Zdroj: vlastní

Termínem „studie“ je v práci myšleno seskupení několika konkrétních měření od různých participantů se stejným zobrazovaným podkladem (obrázek či webová stránka). Každé měření má definovanou svou studii. Případem užití „*Vytvoření nové studie*“ je tedy uvažováno především definování atributů nové studie:

- Název studie
- Délka měření (může být i neomezená – ukončí ji uživatel)
- Typ studie (testování webu či obrázku)
- Zadání URL adresy anebo nahrání testovaného obrázku (dle typu studie)
- Zvolení požadovaných výstupů studie (heat mapa, gaze plot, videozáznam z konkrétních testování)

Případem užití „*Realizace měření*“ se rozumí samotné spuštění nového měření. Konkrétně uživatel vybere, jaká studie má být realizována a jaká data se mají nahrávat (data z oční kamery či myši, snímky z videokamery, zvuk z mikrofonu). Poté následuje samotné spuštění měření, nahrávání a uložení naměřených dat. Spustit nové měření může buď výzkumník, anebo ho může zahájit participant sám.

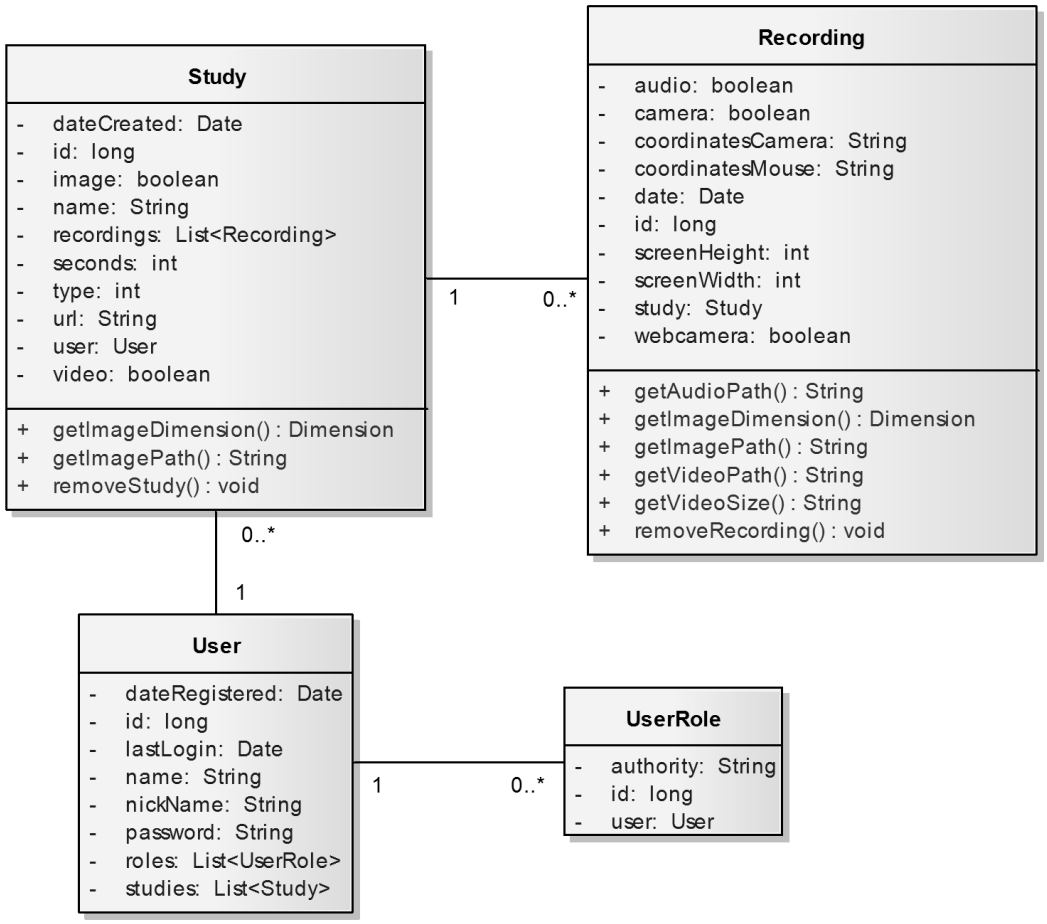
Případem užití „*Zobrazení detailu studie*“ je míněno zobrazení parametrů studie (co je testováno, jak dlouhý časový úsek, co je výstupem apod.), vyobrazení seznamu konkrétních měření této studie a zobrazení heat mapy z agregovaných dat všech měření v této studii.

Případem užití „*Zobrazení detailu konkrétního měření*“ je myšleno především zobrazení vizualizačních technik gaze plot a heatmap vygenerovaných na základě dat tohoto konkrétního měření. Pokud byl při testování nahráván i videozáznam, tak také možnost přehrání tohoto záznamu.

Kromě aktérů výzkumník a participant je v systému nutné počítat i s aktérem administrátor, který si může navíc zobrazit, jací další uživatelé systém používají a případně jim také udělit administrátorská práva.

5.2 Diagram tříd

Z požadavků a následně vytvořeného use case diagramu byly navrženy modelové třídy vytvářené aplikace. Obr. 13 znázorňuje vytvořený model tříd (pro přehlednost nejsou v diagramu znázorněny gettery a settery tříd). Hlavní logiku aplikace pokrývají 4 třídy – *Study*, *Recording*, *User* a *UserRole*.



Obr. 13: Diagram tříd

Zdroj: vlastní

Třídy *User* a *UserRole* reprezentují uživatele a jeho uživatelskou roli v systému. Každý uživatel má atributy, které uchovávají informace o tom, kdy byl registrován, kdy se naposledy přihlásil, jeho přezdívku, heslo a unikátní jméno. Každý uživatel může mít hned několik uživatelských rolí. Každému uživateli také náleží několik studií, které si v systému vytvoří.

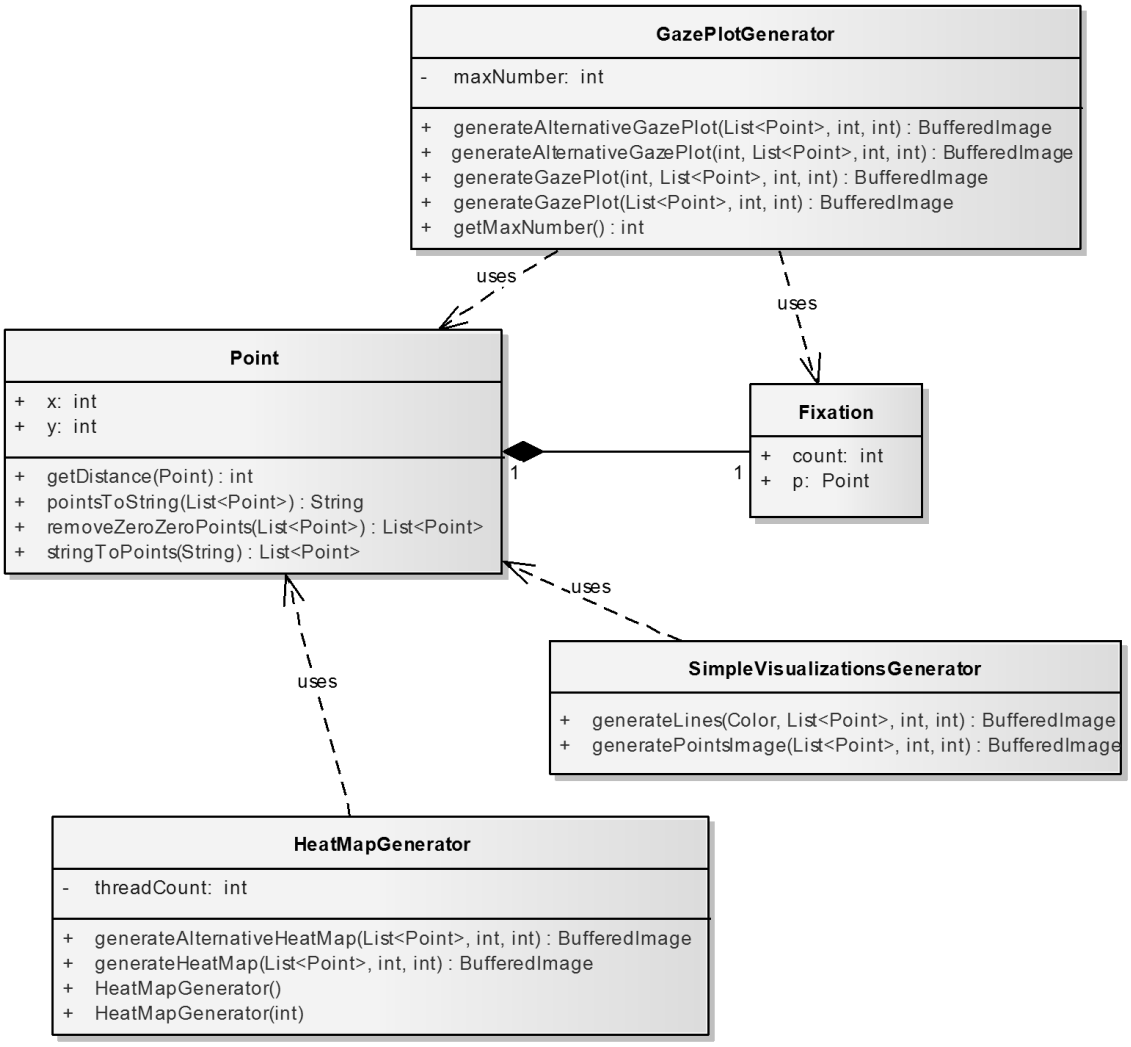
Třída *Study* představuje studii. Každá studie má definováno jméno, datum a čas, kdy byla vytvořena, typ studie (testování webu či obrázku) a délku měření v sekundách. Další atributy definují, zda má být výsledkem studie obrázek (gaze plot a heat mapa) či videozáznam. V případě, že je testován web, je v systému ke studii ukládána i jeho URL adresa. Pokud je ve studii testován obrázek, vrací metoda `getImageDimension` rozlišení testovaného obrázku a metoda `getImagePath`

jeho cestu na disku. Metoda `removeStudy` maže všechny soubory na disku, které náleží k dané studii a také všechny soubory náležící všem měřením této studie.

Objekty třídy *Recording* reprezentují dané měření jednoho participanta. Atribut `coordinatesCamera` představuje nahraná data z oční kamery, `coordinatesMouse` data z pohybu myši. Booleovské atributy `audio`, `camera` a `webcamera` vypovídají o tom, zda byl při měření pro účely tvorby videozáznamu zaznamenáván zvuk z mikrofonu, data z oční kamery anebo snímky z webkamery. V atributech `screenHeight` a `screenWidth` jsou uloženy informace o rozlišení monitoru, na kterém bylo prováděno měření. Metoda `removeRecording` maže všechny soubory na disku, které náleží k danému měření. Při testování webových stránek je ke každému měření vytvářen obrázek – náhled webu. V tomto případě jsou ve třídě k dispozici metoda `getImagePath` a `getImageDimension` s analogickou funkcionalitou jako ve třídě *Study*. Podobnou funkčnost mají i metody `getVideoPath` a `getAudioPath`. Metoda `getVideoSize` vrací velikost vytvořeného videozáznamu v bytech.

5.2.1 Knihovna pro generování vizualizací

Pro účely generování výstupů vizualizačních technik (heat mapa, gaze plot) byla v rámci práce navržena vlastní knihovna. Princip algoritmů vykreslování je v práci popsán níže (6.5). Zde je popsána pouze struktura tříd této knihovny a jejich účel. Navržené třídy zobrazuje Obr. 14.



Obr. 14: Diagram tříd - knihovna pro generování vizualizací

Zdroj: vlastní

Ústřední třídou je třída *Point*. Ta reprezentuje bod se souřadnicemi x a y. Třída obsahuje několik metod:

- Metoda `removeZeroZeroPoints` vznikla především díky využívání oční kamery The Eye Tribe. Pokud tato kamera nerozezná, kam se uživatel dívá, vrací souřadnice [0, 0]. Tato metoda tak vrací seznam naměřených bodů bez takovýchto nežádoucích bodů.
- `getDistance` vrací vzdálenost dvou bodů.
- Metoda `poinstToString` slouží pro převedení seznamu instancí třídy Point do textového řetězce a to tak, že jsou souřadnice odděleny mezerou (např.:

„*120 150 122 152 130 160*“). Metoda `stringToPoints` má opačnou funkcionalitu.

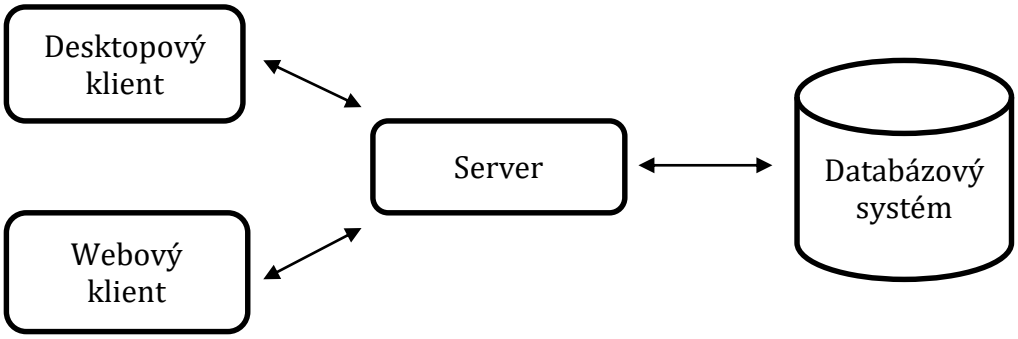
Třída *HeatMapGenerator* slouží k vytváření heat map. Má dva konstruktory – s celočíselným parametrem typu `int` a bez parametru. Parametr konstruktoru určuje, kolik vláken se má při výpočtu použít (konkrétněji je algoritmus popsán v odstavci 6.5.1). Konstruktor bez parametru přiřadí pro výpočet právě jedno vlákno. Metoda `generateHeatMap` vrací vygenerovanou heat mapu jako obrázek typu `BufferedImage`, který je vytvořený ze vstupních parametrů – požadované šířky a výšky výstupu a seznamu bodů. Metoda `generateAlternativeHeatMap` má stejnou funkcionalitu, pouze generuje heat mapu s alternativní barevnou stupnicí (fialovo-červeno-žlutá).

Účelem třídy *GazePlotGenerator* je generování vizualizace gaze plot. Metoda `generateGazePlot` vrací vygenerovaný obrázek typu `BufferedImage`. Vstupem metody je také požadovaná šířka a výška výstupu a seznam bodů. Vstupem metody může být i počáteční index, tedy číslo, které je vykreslované dovnitř kruhů znázorňujících fixace – defaultně je indexem číslo 1. Metoda `generateGazePlot` vykresluje gaze plot černou barvou (viz Obr. 22), `generateAlternativeGazePlot` barvou fialovou. Pomocí `getMaxNumber` lze po vygenerování získat nejvyšší index, jaký byl při generování použit. Třída *GazePlotGenerator* používá při výpočtech pomocnou třídu *Fixation*, reprezentující fixaci pohledu.

Poslední třídou knihovny je *SimpleVisualizationsGenerator*. Ta dokáže generovat obrázek, ve kterém jsou souřadnice x, y zadaných bodů vykresleny černě formou teček na bílém podkladě (metoda `generatePointsImage`). Také dovede vykreslit lomenou čáru spojující body ze vstupního seznamu (metoda `generateLines`).

5.3 Architektura aplikace

Pro tvorbu softwaru k vyhodnocování dat z uživatelského testování s oční kamerou The Eye Tribe byla zvolena vícevrstvá architektura, jejíž hlavní části jsou vyobrazeny na Obr. 15.



Obr. 15: Architektura aplikace

Zdroj: vlastní

Klientská část se skládá jak z webového, tak i desktopového klienta. Desktopový klient byl použit především kvůli nutnosti spolupracovat s hardwarem počítače (oční kamera, webkamera, mikrofon apod.).

Webový klient slouží zejména pro analýzu naměřených dat, konkrétně:

- registraci do aplikace,
- zobrazení celkového přehledu – počet vytvořených studií a měření, datum posledního měření, seznam posledních provedených měření,
- založení nové studie – viz případ užití „*Vytvoření nové studie*“ (5.1),
- zobrazení detailu dané studie – viz případ užití „*Zobrazení detailu studie*“ (5.1),
- zobrazení detailu konkrétního měření – viz případ užití „*Zobrazení detailu konkrétního měření*“ (5.1).

Úlohou desktopového klienta je spuštění nahrávání nového měření – viz případ užití „*Realizace měření*“ (5.1). Zaznamenaná data poté desktopový klient odesílá na server, kde jsou zpracována a poskytnuta uživateli skrze webového klienta.

Takováto architektura aplikace byla zvolena především pro možnost centralizace naměřených dat na straně serveru. Dalším důvodem je, že na klientský počítač nejsou kladeny tak vysoké výpočetní nároky – výpočty (např. generování heat map) jsou přesunuty na server. Pro analýzu dat není nutné na klientský počítač

instalovat další aplikaci, práce probíhá přes webový prohlížeč. Desktopový klient je používán pouze pro vytváření nových měření.

Typický postup při využívání aplikace se dá popsat několika kroky:

1) Pokud uživatel ještě nemá svůj účet, zaregistruje se pomocí webového klienta.
2) Po přihlášení ve webovém klientovi si vytvoří novou studii s požadovanými parametry (délka měření, typ studie atd.).
3) Pokud již má uživatel vytvořenou alespoň jednu studii, spustí desktopového klienta, kde se mu zobrazí seznam jeho studií, jednu z nich vybere, zvolí, co vše se má nahrávat, a spustí nahrávání.
4) Po ukončení nahrávání jsou všechna data nahrána na server a uživatel je může pomocí webového klienta analyzovat.

6 Implementace

V této části práce jsou zmíněny implementační detaily navrženého softwaru. Je zde popsána serverová, datová a klientská část aplikace. Jako samostatná podkapitola je vyčleněn popis implementace algoritmů pro vykreslení vizualizací gaze plot a heatmap.

6.1 Serverová část

Většina funkcionality celé aplikace je umístěna na straně serveru. Serverová část požaduje pro svůj běh webový server s podporou servletového kontejneru. Aplikace byla konkrétně testována na serveru Apache Tomcat.

Serverová část je postavena na frameworku Spring. Pro správu závislostí na konkrétních knihovnách, které aplikace pro svůj běh vyžaduje, je využit nástroj Maven. Ten je použit i v desktopovém klientu.

Objektově relační mapování obstarává v aplikaci nástroj Hibernate – třídy jsou mapovány na databázové tabulky pomocí JPA anotací. Díky tomuto nástroji není nutné definovat ručně databázové tabulky, ty jsou vygenerovány automaticky z vytvořených Java tříd.

Pro rychlejší vývoj byl z počátku využíván nástroj Spring Roo. Ten je využíván i ve finální verzi aplikace, a to především pro zpřehlednění kódu modelových tříd, ve kterých není nutné definovat gettery, settery a některé další atributy a metody, které vygeneruje tento nástroj sám.

Logování je na straně serveru i v desktopovém klientu realizováno za pomoci nástroje Log4j. Nastavení logování lze měnit v samostatném souboru *log4j.properties*.

6.2 Datová část

Pro persistenci dat byl zvolen databázový systém MySQL – především pro svou jednoduchost a také předchozí zkušenosti autora. Jelikož je v práci použit

objektově relační framework Hibernate, přechod na jiný databázový systém by neměl být velkým problémem.

6.3 Klientská část

Jak již bylo zmíněno výše (5.3), klientská část aplikace se skládá z webového a desktopového klienta.

6.3.1 Desktopový klient

Pro tvorbu grafického rozhraní desktopového klienta byl použit nástroj JavaFX. Struktura aplikace je popsána pomocí FXML souborů, vzhled je upravován pomocí CSS.

Kromě standardních úkolů desktopového klienta (např. přihlášení, zobrazení seznamu studií apod.) bylo v rámci práce řešeno i několik specifických problémů pro záznam uživatelského testování s oční kamerou, které budou dále popsány.

6.3.1.1 Testování obrázků

Jedním z typů studie, které lze vytvořit, je testování statického obrázku. Ve vytvořené aplikaci je tento obrázek nahrán na server již při zakládání nové studie. Při testování pomocí desktopového klienta je tento obrázek nejprve stažen desktopovým klientem ze serveru. K jeho zobrazení je poté využita třída `ImageView` z balíčku `javafx.scene.image`.

Obrázek je zobrazen přes celou obrazovku monitoru. Jestliže je obrázek menší než rozlišení monitoru, je umístěn na střed. Pokud je větší, je uživateli umožněno scrollování. Zde nastává problém při generování heat mapy a gaze plot – data poskytnutá z oční kamery jsou uváděna v soustavě souřadnic monitoru. Pro generování je však nutné znát tato data v soustavě souřadnic obrázku. Je tak nutné tyto souřadnice převést do soustavy souřadnic zobrazovaného obrázku. Je-li obrázek menší než rozlišení monitoru, lze tento výpočet jednoduše realizovat na základě šířky (popř. výšky) obrázku a rozlišení monitoru. Naopak, je-li obrázek větší než rozlišení monitoru a uživatel scrolluje, je nutné zjistit, o kolik pixelů je scrollováno (horizontálně či vertikálně). K tomu se dá využít třída `ScrollPane`

z balíčku `javafx.scene.control`, kterou je obrázek (ImageView) „obalen“. Třída `ScrollPane` poskytuje pro tyto účely svoje metody `getHvalue`, `getVvalue`, `getHmax` a `getVmax`.

Konkrétně pro převedení vertikální souřadnice („y“) může sloužit následující výňatek zdrojového kódu:

```
public int getScrollTop() {
       int paddingTop = 0;
       if (img.getHeight() < screenHeight) {
           paddingTop = (int) ((screenHeight - img.getHeight()) / 2);
       }
       int scroll =
            (int) ((img.getHeight() - scrollPane.getHeight()) *
            (scrollPane.getVvalue() / scrollPane.getVmax()));
       scroll = scroll - paddingTop;
       return scroll;
}
```

Výpočet pro horizontální souřadnice je analogický.

6.3.1.2 Testování webů

Jednou z možností, jak testovat webové stránky, je vytvořit náhled celého webu (snapshot) na serveru a následně zobrazit participantovi tuto webovou stránku formou statického obrázku. Je tak zaručeno, že je vždy participantům zobrazován stejný podklad. Nevýhodou je, že se nejedná o „reálnou“ webovou stránku, a není tak možné participantovo „reálné“ chování (např. proklikávání na jinou stránku). Tento způsob používá aplikace EyeProof (3.4).

V této práci byl však využit jiný přístup. Participantovi je při testování zobrazena „reálná“ webová stránka a snímek webu je vytvořen s každým konkrétním měřením (dle toho, co je participantovi zobrazováno). Jedním ze způsobů, jak tuto funkcionalitu implementovat, je využití JavaFX třídy `WebView` z balíčku `javafx.scene.web`. Ze zkušeností autora byl však tento způsob zavržen z toho důvodu, že ne všechny testované weby byly zobrazeny stejně jako v nejpoužívanějších webových prohlížečích. Proto byl využit prohlížeč Mozilla Firefox ve spolupráci s nástrojem Selenium. I když není Selenium primárně určeno pro toto využití, jeho funkcionalitu lze tímto způsobem využít. Tento nástroj

dokáže vytvořit snímek celého webu (např. o velkosti 1920 × 3000), který je právě participantovi zobrazován.

Stejně jako u testování obrázků, je nutné zjistit o kolik pixelů je v daný okamžik webová stránka scrollovaná. K tomu lze využít javascriptových vlastností `scrollTop` a `scrollLeft`, které udávají, o kolik pixelů je daný objekt „scrollován“ vertikálně a horizontálně. Konkrétně je tato funkcionalita v aplikaci implementována pomocí nástroje Selenium následovně:

```
public Integer getScrollTop() {
   try {
      WebElement element = driver.findElement(By.tagName("html"));
      Long l = (Long) (driver.executeScript(
            "return arguments[0].scrollTop;", element));
      return l.intValue();
   } catch (Exception e) {
      return null;
   }
}
```

Nevýhoda tohoto přístupu oproti tomu použitém v EyeProof je, že na klientském počítači musí být dostupný prohlížeč Mozilla Firefox (buď klasicky nainstalován, anebo alespoň složka s portable verzí). Další nevýhodou je, že není zaručen stejný vzhled webové stránky při každém dílčím měření, což je podstatným předpokladem např. pro generování heat mapy z agregovaných dat ze všech měření dané studie. V tomto případě je na výzkumníkovi, aby zajistil, že bude daná stránka vždy vypadat stejně.

I přes nevýhody je tento přístup použit především díky jistotě, že bude web vypadat stejně jako v nejpoužívanějších prohlížečích, což Mozilla Firefox bezesporu je. Za další výhodu lze považovat to, že je participantovi zobrazován „reálný web“, ne pouze obrázek webu.

6.3.1.3 Kalibrace oční kamery

Pro správnou funkčnost oční kamery The Eye Tribe je nutné před používáním spustit aplikaci EyeTribe Server. Pro usnadnění práce je v aplikaci

implementováno automatické spouštění této aplikace na pozadí (funkční pouze na operačních systémech Microsoft Windows při defaultním umístění instalace).

Pro další usnadnění práce s aplikací je implementována i vlastní kalibrace oční kamery, uživatel tak nemusí pro kalibraci spouštět EyeTribe UI. Možnost kalibrace kamery na platformě Java byla přidána v listopadu roku 2015 do knihovny „*The Eye Tribe Java SDK*“ (4.7). Je tak využito funkcionalit této knihovny pro kalibraci zařízení.

6.3.1.4 Tvorba videozáznamu z uživatelského testování

Jednou z možností výstupu aplikace je i videozáznam z konkrétního uživatelského testování. K těmto účelům je v desktopovém klientu použit nástroj Xuggler popsaný výše (4.6).

K vytváření videí jsou používány snímky obrazovky, které lze v Javě získat pomocí třídy `java.awt.Robot` a její metody `createScreenCapture`. Takovýto snímek je však zachycen bez vyobrazení kurzoru myši. Ten je proto do snímku přidán později ve formě obrázku. Do snímků je následně vykresleno i místo, kam v daný časový okamžik směřoval pohled uživatele. Je vyobrazena i trajektorie pohledu z několika posledních okamžiků. Pokud uživatel zvolí, že se mají nahrávat i data z webkamery, do pravého spodního rohu snímku je přidán i pořízený snímek. Pro práci s webkamerou je v práci využita knihovna s názvem Webcam Capture API, která je dostupná na serveru GitHub [25]. Při práci s webkamerou nastávaly při vývoji aplikace problémy, jelikož se oční kamera The Eye Tribe jeví této knihovně také jako webkamera. Tento problém byl vyřešen s využitím třídy knihovny `Webcam` a její metody `getName`.

Před zahájením měření je v desktopovém klientu možné zvolit i tvorbu audiozáznamu z uživatelského testování. K vytváření audiozáznamu z mikrofonu bylo využito tříd z balíčku `javax.sound.sampled`. Audiozáznam je pak uložen a odeslán na server jako soubor s příponou *wav*.

6.3.1.5 Odesílání pořízených měření na server

Jestliže uživatel zahájí měření pomocí desktopového klienta a poté ho ukončí, jsou nejprve uložena všechna naměřená data na pevný disk. Následně je nutné tato data odeslat na server. Odesílání dat však může být dlouhotrvající proces (především v případě, že je vytvářen několikaminutový videozáznam). Z toho důvodu je po ukončení měření uživateli nabídnuto, zda chce tato data odeslat ihned anebo později. V případě, že zvolí druhou možnost, jsou data ponechána na disku a uživatel je může odeslat kdykoli zvolí za vhodné. Po odeslání jsou data smazána z disku.

6.3.1.6 Komunikace klienta se serverem

Desktopový klient komunikuje se serverem pomocí zasílání HTTP (Hypertext Transfer Protocol) zpráv. Pro implementaci byly použity třídy projektu Apache HttpComponents [1] . Při komunikaci je využíván formát JSON.

6.3.2 Webový klient

Webový klient je implementován s využitím nástroje Spring MVC, který je založen na návrhovém vzoru Model-View-Controller. Kontroléry jsou realizovány pomocí tříd s anotací @Controller. View (pohled) je řešeno JSP stránkami s využitím značek knihovny JSTL.

6.3.2.1 Generování vizualizací heatmap a gaze plot

Ke generování vizualizací heatmap a gaze plot je v aplikaci využívaná vlastní knihovna zmíněná výše (5.2.1). Ta generuje obrázky s průhledným pozadím. Ve webovém klientu je vždy zobrazen podklad, který při testování viděl participant, jako klasický obrázek (HTML značka <img>). Ten je pomocí CSS a absolutního pozicování překryt např. vygenerovaným obrázkem heat mapy. Co vše se má zobrazit, určuje uživatel pomocí checkboxů, defaultně je zobrazena heat mapa vygenerovaná z dat naměřených oční kamerou. Kromě dat z oční kamery nahrává aplikace i pozici myši na monitoru. Tato data lze také využít pro generování zmíněných vizualizací – zajímavá je především možnost vygenerování techniky „gaze plot“, která zobrazuje, jak se pohybovala myš po monitoru v průběhu měření.

Součástí aplikace je i možnost exportu. Uživatel tak může stáhnout obrázek ve formátu JPEG, na kterém je vždy vyobrazen podklad, který byl zobrazován participantovi při testování, a na něm nanesena např. heat mapa či gaze plot – podle požadavků uživatele.

6.3.2.2 Přehrání pořízeného videozáznamu

K přehrání videa je využita značka <video> ze specifikace HTML 5. Jelikož je desktopovým klientem tvořen videozáznam a audiozáznam zvlášť do samostatných souborů, je nutné je při přehrání spojit a synchronizovat. Ve webovém klientu je za tímto účelem přidán i audiozáznam pomocí značky <audio>. Synchronizace audia s videem je poté realizována pomocí JavaScriptu – audio a video je spuštěno a přehráváno zároveň.

Důležité pro uživatele je i stažení videa na lokální disk. Zde je nutné audio s videem spojit do jednoho souboru. Původně byl využit v aplikaci za tímto účelem nástroj Xuggler, pro svou výpočetní náročnost byl však nahrazen nástrojem FFmpeg (popsán výše – 4.6). Spojení audia s videem lze pomocí FFmpeg realizovat následovně:

```
ffmpeg -i audioInput.wav -i videoInput.avi -acodec copy -vcodec copy
outputFile.avi
```

6.4 Zabezpečení

Důležitou součástí aplikace je její zabezpečení. Pro využívání aplikace je nutné vytvoření uživatelského účtu. Jako jednoznačný identifikátor slouží emailová adresa uživatele. Hesla jsou uložena v databázi v hashované podobě.

6.4.1 Zabezpečení na straně serveru

Na straně serveru je zabezpečení řešeno nástrojem Spring Security, který poskytuje prostředky pro snadnou autentizaci a autorizaci uživatelů. Konfigurace tohoto nástroje je oddělena do samostatného souboru spring-security.xml.

Zabezpečení probíhá již na úrovni URL adresy:

- Úvodní stránka (tj. přihlašovací formulář) a registrační formulář (/registration) jsou dostupné každému.
- Všechny ostatní URL adresy jsou zpřístupněny pouze přihlášeným uživatelům.
- Všechna URL začínající na <adresa_serveru>/admin/ jsou dostupná pouze uživatelům s právy administrátora.

Dále je na straně serveru řešeno zabezpečení vytvořených studií a měření. Studie a měření jsou dostupné pouze jejich vlastníkovi. To je v aplikaci implementováno s využitím aspektově orientovaného programování – oprávnění uživatele je ověřeno ještě před voláním metody kontroléru.

6.4.2 Zabezpečení na straně desktopového klienta

Pro přihlášení na straně desktopového klienta slouží formulář pro zadání emailové adresy uživatele a jeho hesla. Pro ověření, zda zadaný email a heslo odpovídají, je odesílán HTTP požadavek na server. Server poté tento požadavek zpracuje a vrací příslušnou odpověď.

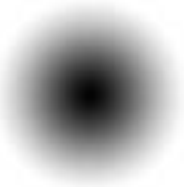
6.5 Implementace vizualizačních algoritmů

Jak již bylo zmíněno výše (5.2.1), v rámci práce byla pro účely generování vizualizací navržena vlastní knihovna v programovacím jazyce Java. V této části práce jsou popsány algoritmy vykreslení vizualizací heatmap a gaze plot a jejich implementace.

6.5.1 Algoritmus vykreslení heat mapy

Obecný princip algoritmů tvorby heat map, které využívají softwary pro eye tracking, je stručně nastíněn v knize [6]. Na stejném principu je založen i jednoduchý algoritmus uvedený na webu [13] implementovaný v jazyce Java.

Vstupem algoritmu je množina bodů se souřadnicemi x, y. Výstupem pak dvourozměrný obraz s vykreslenou heat mapou. Základem algoritmu [13] je využití podobného obrázku, jaký znázorňuje Obr. 16.



Obr. 16: Obrázek, který využívá algoritmus vykreslení heat mapy

Zdroj: vlastní

Tento dvourozměrný obrázek je postupně nanášen na statický podklad na souřadnice bodů (x, y) ze vstupu. Podklad je v algoritmu [13] implementován jako plně průhledný obraz pomocí třídy jazyka Java `BufferedImage` z balíku `java.awt.image`. Po nanesení obrázku na všechny souřadnice bodů x, y vzniká pomocný obraz, který může vypadat podobně jako Obr. 17.

Obr. 17: Pomocný obraz při vytváření heat mapy

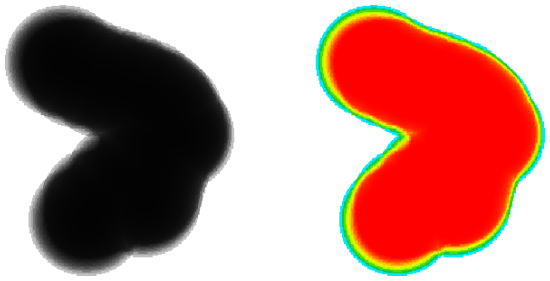
Zdroj: vlastní

Z takovéhoto pomocného obrazu lze již vygenerovat obrázek heat mapy pomocí jednoduchého přemapování barev. Nejvíce průhledná místa jsou obarvena typicky modře a nejméně průhledná červeně. Barvy ostatních oblastí lze dopočítat lineární interpolací s využitím stupnice barev, např. takové, jaká je znázorněna na Obr. 18.

Obr. 18: Stupnice barev od studených po teplé

Zdroj: vlastní

Algoritmus [13] je sice funkční, avšak má několik nedostatků. První z nich nastává při generování heat mapy z velkého množství zadaných bodů. Pokud jsou body poblíž sebe a nanášené obrázky (Obr. 16) se tak postupně překrývají, naráží tento algoritmus na omezené rozpětí hodnot alfa kanálu barev. Hodnota alfa kanálu tak dosáhne své mezní hodnoty a mohou vznikat např. obrazce, jaké jsou znázorněny na Obr. 19 (vlevo obrazec při generování pomocného obrazu, vpravo již po přemapování barev).



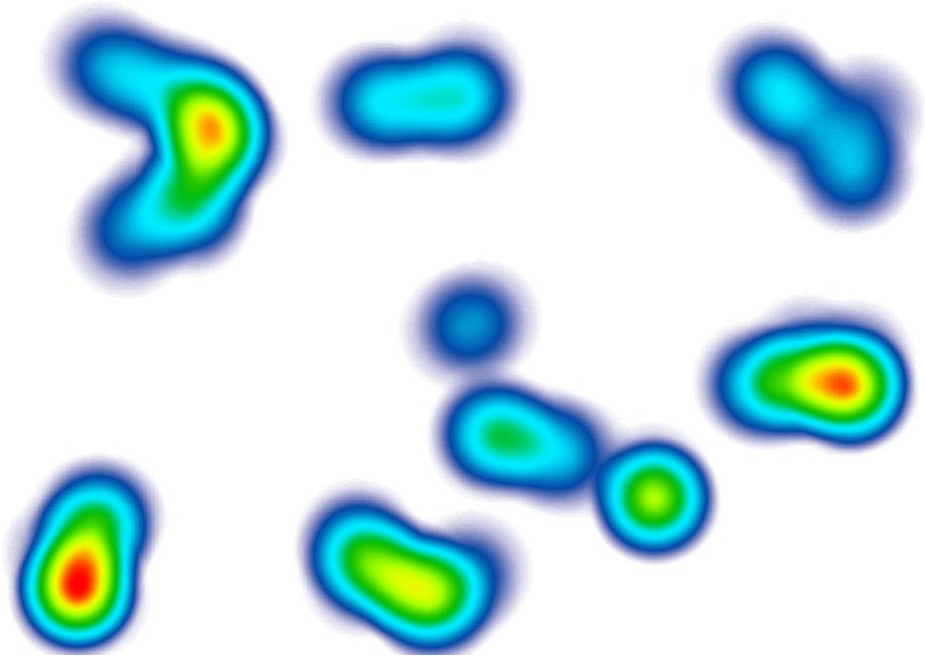
Obr. 19: Chyba algoritmu při velkém množství vstupních bodů

Zdroj: vlastní

Dalším nedostatkem algoritmu [13] je, že neumí pracovat paralelně ve více vláknech.

Ze zmíněných důvodu byl v práci implementován vlastní algoritmus pro tvorbu heat mapy. Ten vychází ze stejných principů, jaké jsou zmíněny ve zdrojích [6] a [13]. Základem je obrázek podobný Obr. 16. Jeho velikost je automaticky měněna podle velikosti podkladu (např. webová stránka, na kterou se participant díval). Z tohoto obrázku je vytvořeno dvourozměrné pole typu `short[][]` stejné velikosti, jakou má tento obrázek. Hodnoty pole jsou přepsány tak, aby odpovídaly hodnotám alfa kanálů načtených z obrázku.

Dále je vytvořeno další dvourozměrné pole typu `int[][]`, které má stejnou velikost jako podklad zobrazovaný participantovi. Hodnoty z menšího pole (`short[][]`) jsou poté postupně přičítány k hodnotám velkého pole (`int[][]`) na místech bodů (x, y) ze vstupu. Oproti algoritmu [13] tak není pracováno se třídou `BufferedImage`, ale s dvourozměrnými poli. Je tak zaručeno mnohem vyšší rozpětí hodnot, které lze v poli ukládat, a i pro velká množství bodů je toto rozpětí dostatečné. Následně jsou hodnoty ve velkém poli převedeny na celočíselné hodnoty v rozmezí 0 až 255 (odpovídající hodnotám alfa kanálů), ze kterých je vygenerován obrázek typu `BufferedImage` podobný Obr. 17. Dále je provedeno přemapování barev stejně, jak již bylo zmíněno výše. Navržený algoritmus navíc upravuje alfa kanál hodnotám s nejnižší intenzitou (typicky obarveny modře) – výsledkem pak je, že modré oblasti přecházejí plynule do průhledných. Výsledná heat mapa vygenerovaná navrženým algoritmem pak může vypadat např. jako na Obr. 20.



Obr. 20: Heat mapa vygenerovaná navrženým algoritmem

Zdroj: vlastní

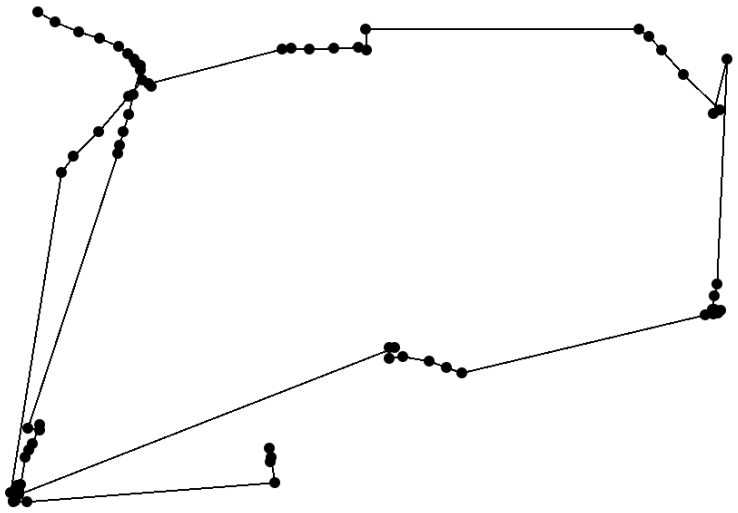
V navrženém algoritmu byla implementována i možnost paralelního zpracování. Vstupem algoritmu pak je i počet vláken, v kolika má výpočet probíhat. Množina všech vstupních bodů je poté rozdělena na přibližně stejně velké podmnožiny

(počet vláken odpovídá počtu podmnožin), které jsou zpracovávány paralelně. Vzniká tak hned několik stejně velkých dvourozměrných polí typu `int[][]`, jejichž hodnoty na stejných pozicích jsou posčítány a součty uloženy do nového, stejně velkého pole. To je následně převedeno na obrázek typu `BufferedImage` podobný Obr. 17 a následující postup je již identický s tím, který byl zmíněn výše.

Nevýhodou tohoto paralelního zpracování jsou vyšší paměťové nároky, především z nutnosti používání hned několika dvourozměrných polí typu `int[][]`. Při dostatečných paměťových prostředcích je však možné dosáhnout několikanásobně rychlejšího výpočtu.

6.5.2 Algoritmus vykreslení vizualizační techniky gaze plot

Vstupem algoritmu pro vykreslení vizualizační techniky gaze plot je stejně jako u vykreslování heat mapy množina bodů se souřadnicemi x, y. Navíc je nutné znát i jejich pořadí, v jakém následují po sobě. Jak již bylo zmíněno v odstavci 2.9.2, jsou fixace pohledu znázorněny jako kruhy s čísly a sakády jako čáry. Zásadním úkolem při vykreslování techniky gaze plot je tedy nalezení fixací v seznamu zadaných bodů (které poskytuje oční kamera). Takovýto seznam bodů pospojovaných čarami podle jejich pořadí může vypadat např. jako na Obr. 21.



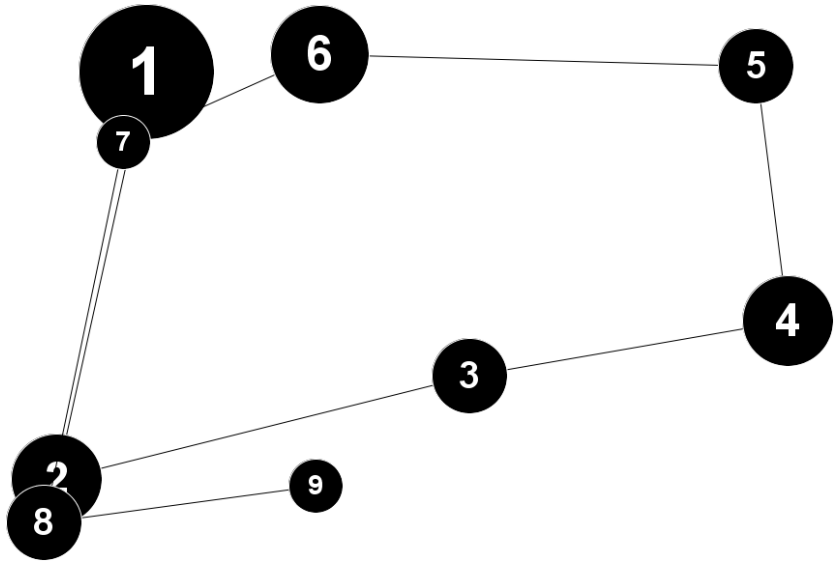
Obr. 21: Množina vstupních bodů a jejich pořadí

Zdroj: vlastní

V rámci práce byl navržen vlastní algoritmus pro vykreslení vizualizační techniky gaze plot. Prvním krokem je nalezení fixací. Ty jsou v algoritmu definovány dvěma údaji – pozice (souřadnice x, y) a významnost (daná počtem bodů, které odpovídají jedné fixaci). Algoritmus vychází z předpokladu, že pokud jsou od sebe dva po sobě následující body dostatečně vzdálené, jedná se o sakádu. Pokud jsou blízko u sebe, dá se předpokládat, že se jedná o tu samou fixaci. Algoritmus tedy prochází postupně všechny body, porovnává jejich vzdálenosti od následujících bodů a hledá tak skupiny bodů oddělené sakádami. Z každé takové nalezené skupiny je poté definována fixace s významností odpovídající počtu bodů ve skupině. Souřadnice této fixace jsou spočteny průměrem souřadnic všech bodů skupiny. Důležité je správně určit, jaká vzdálenost bodů je při výpočtu tohoto algoritmu „dostatečně velká“. Ze zkušeností autora při využívání algoritmu byla zvolena vzdálenost 150 pixelů při šířce podkladu 1200 pixelů, která je však přepočítána v závislosti na šířce podkladu.

Dalším krokem algoritmu je samotné vykreslení kruhů s čísly a spojnic mezi nimi. To je implementováno metodami třídy `Graphics2D` z balíčku `java.awt`. Velikost kruhů a textu je upravována v závislosti na šířce podkladu.

Výsledek algoritmu, jehož vstupem jsou body znázorněné na Obr. 21, zobrazuje Obr. 22.



Obr. 22: Gaze plot vygenerovaný navrženým algoritmem

Zdroj: vlastní

6.5.3 Rychlost algoritmů

V rámci práce byla orientačně otestována i rychlost výpočtu zmíněných algoritmů (6.5.1 a 6.5.2). Hlavní motivací bylo především otestovat, jaký vliv na rychlost výpočtu má počet využitých vláken při generování heat map.

Testování proběhlo na přenosném počítači Dell Vostro 3750 s procesorem Intel Core i7-2630QM (čtyřjádrový procesor s podporou technologie Hyper-threading) a velikostí operační paměti 6GB. Algoritmus byl testován na různém počtu vstupních bodů, konkrétně 100, 1 000, 50 000 a 500 000 vstupních bodů. Oční kamera The Eye Tribe poskytuje data s frekvencí 30 bodů za jednu vteřinu. Generování heat mapy z 500 000 bodů tedy zhruba odpovídá generování heat mapy ze 4,5 hodinového testování s touto kamerou.

K testování rychlosti generování heat mapy byl použit následující kód:

```
int threadCount = 4; // počet vláken použitých při výpočtu
HeatMapGenerator g = new HeatMapGenerator(threadCount);
long startTime = System.currentTimeMillis();
g.generateHeatMap(1920, 1080, points); // samotné spuštění algoritmu
long stopTime = System.currentTimeMillis();
System.out.println(stopTime - startTime); // výpis délky výpočtu
```

Naměřené časy výpočtů v sekundách zobrazuje Tab. 2.

Tab. 2: Rychlost algoritmu generování heat mapy

	Počet vstupních bodů			
	100	**1 000**	**50 000**	**500 000**
1 vlákno	0,61 s	0,85 s	6,03 s	27,53 s
2 vlákna	0,72 s	0,85 s	2,66 s	14,27 s
4 vlákna	0,77 s	0,80 s	1,98 s	8,44 s
8 vláken	0,90 s	0,87 s	1,86 s	7,89 s

Zdroj: vlastní

Jak lze vyčíst z Tab. 2, počet využitých jader při generování heat map má vliv především při větším počtu bodů (v řádu desetitisíců). Algoritmus je poté i několikanásobně rychlejší než při výpočtu v jediném vlákně. Při výpočtech

s malým počtem bodů je režie spojená s používáním několika vláken zbytečně náročná a je lepší využít pouze jednovláknový výpočet.

Rychlost generování vizualizace gaze plot (viz Tab. 3) je spíše informativní, algoritmus neprovádí výpočet ve více vláknech, jako je tomu u generování heat map.

Tab. 3: Rychlost algoritmu generování vizualizace gaze plot

	počet vstupních bodů			
	100	1 000	50 000	500 000
Délka výpočtu	0,09 s	0,15 s	1,68 s	13,82 s

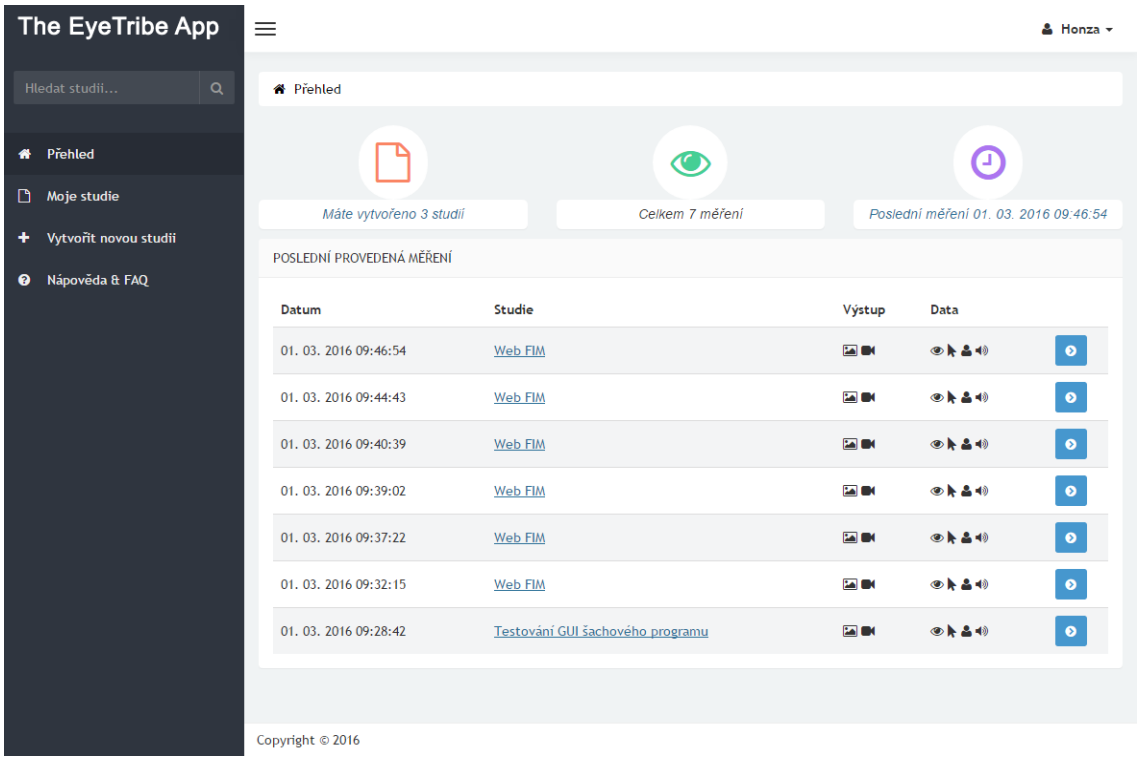
Zdroj: vlastní

7 Výsledky

V práci byl navržen a implementován software k vyhodnocování dat z uživatelského testování s oční kamerou The Eye Tribe. Navržený software lze však využít i pro tvorbu záznamů uživatelských testování bez oční kamery. Pro implementaci byla zvolena platforma Java s vhodnými technologiemi.

Součástí architektury softwaru je jak desktopový, tak i webový klient. Desktopový klient slouží především pro nahrávání dat z testování a jejich následné odeslání na server. Hlavní úlohou serveru a webového klienta je tato data zpracovat a zprostředkovat výsledky uživateli.

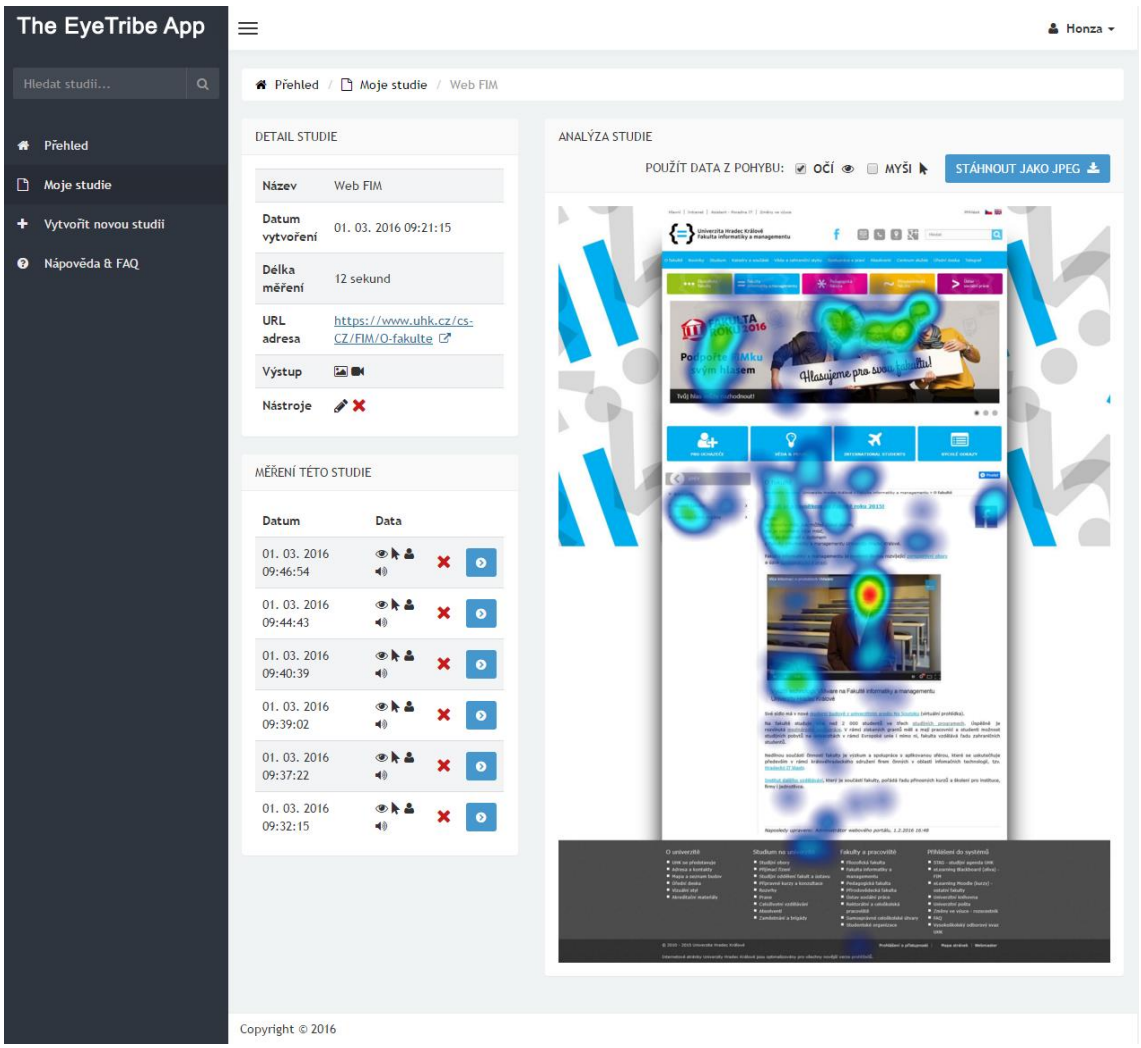
Grafické rozhraní webového klienta zobrazuje Obr. 23.



Obr. 23: Snímek obrazovky webového klienta – přehled

Zdroj: vlastní

Jednou z nejdůležitějších funkcionalit webového klienta je zobrazení heat mapy z agregovaných dat všech měření dané studie. Obr. 24 zachycuje snímek aplikace, ve kterém je zobrazena vygenerovaná heat mapa z celkem šesti dvanáctivteřinových měření, ve kterých byla participantovi vždy zobrazena webová stránka *https://www.uhk.cz/cs-CZ/FIM/O-fakulte*.



Obr. 24: Snímek obrazovky webového klienta – detail studie

Zdroj: vlastní

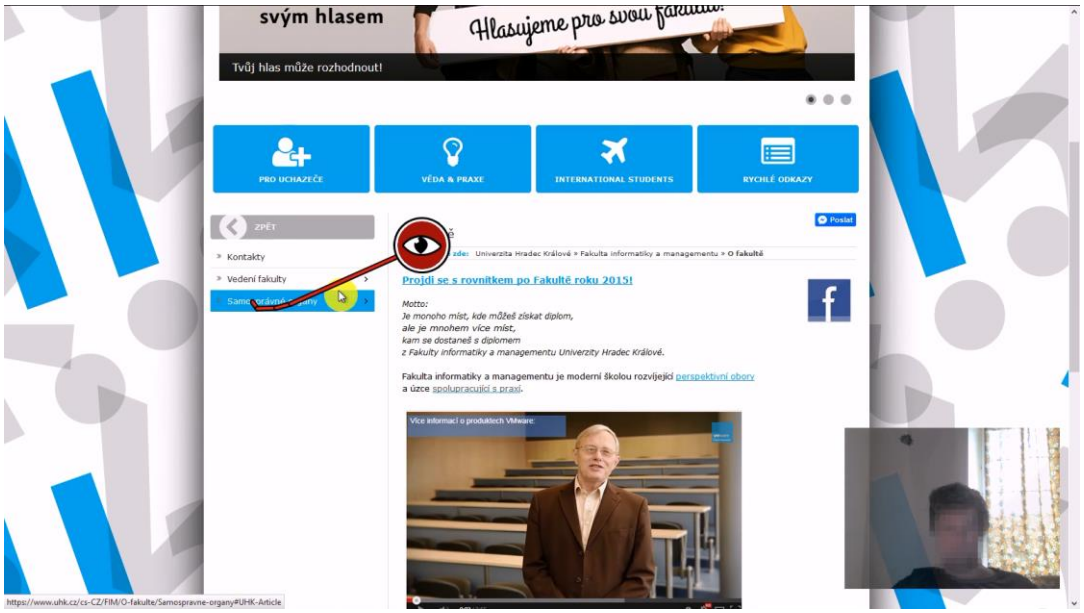
Další (neméně důležitá) funkcionalita webového klienta je vyobrazení detailu konkrétního měření jednoho participanta. Výstupem aplikace pak může být např. vygenerovaný obrázek, ve kterém je znázorněna jak heat mapa, tak i gaze plot, popřípadě i vyobrazení trajektorie pohybu myši v průběhu testování. Takovýto výstup může vypadat např. jako na Obr. 25.

Obr. 25: Export konkrétního měření – heat mapa a gaze plot

Zdroj: vlastní

Kromě exportu obrázku z daného měření může být výstupem i videozáznam z provedeného uživatelského testování. Videozáznam je vytvářen ze snímků obrazovky. Do těchto snímků mohou být vykresleny také místa, kam se uživatel aktuálně dívá, trajektorie pohledu anebo snímky z webkamery. Do videozáznamu

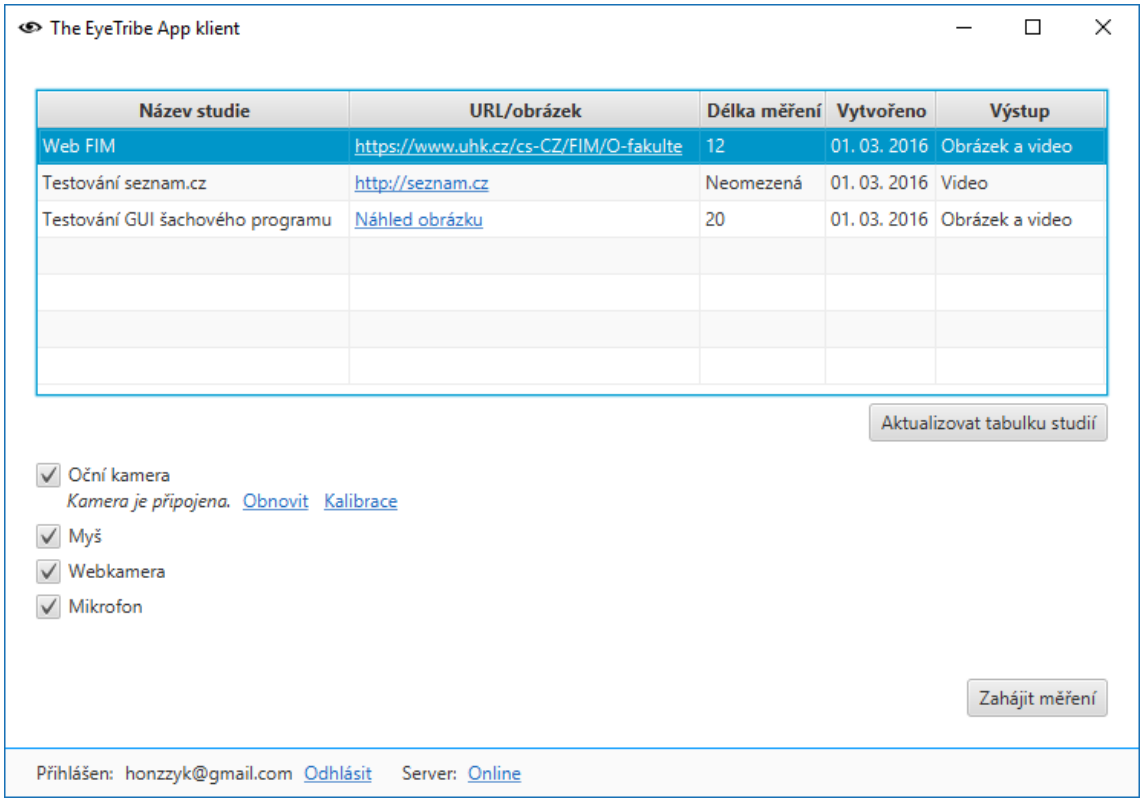
může být přidán i audiozáznam z mikrofonu. Ukázkový snímek z takového videozáznamu je vyobrazen na Obr. 26.



Obr. 26: Snímek videozáznamu z testování

Zdroj: vlastní

Grafické rozhraní desktopového klienta znázorňuje Obr. 27.



Obr. 27: Snímek obrazovky desktopového klienta

Zdroj: vlastní

V rámci práce byla vytvořena samostatná knihovna v jazyce Java, která slouží především pro generování vizualizací heatmap a gaze plot. Pro generování výstupů těchto vizualizačních technik byly navrženy a implementovány vlastní algoritmy. Algoritmus generování heat map dokáže výpočet realizovat ve více vláknech. Vícevláknový výpočet je oproti jednovláknovému při větším množství vstupních bodů až několikanásobně rychlejší (viz Tab. 2).

Aplikaci v současné době (březen 2016) využívá UX agentura CIRCUS DESIGN s.r.o. ke svým interním účelům.

8 Závěr

Cílem práce bylo seznámit se s problematikou uživatelského testování prostřednictvím oční kamery a navrhnout a implementovat software k vyhodnocování dat z uživatelského testování s oční kamerou.

První část práce byla zaměřena na problematiku sledování pohybu očí. Byly popsány principy, na kterých pracují oční kamery, a také zmíněny různé druhy těchto zařízení. Obecně byly popsány i vizualizační techniky, které lze pro účely sledování pohybu očí využít. Práce obsahovala i detailnější popis oční kamery The Eye Tribe včetně programového API, pomocí kterého lze s tímto zařízením pracovat.

Součástí práce byla analýza, návrh a implementace softwaru k vyhodnocování dat z uživatelského testování s oční kamerou The Eye Tribe. Byly zmíněny a stručně popsány technologie, které byly využity při tvorbě této aplikace. Pro implementaci byla zvolena vícevrstvá architektura s webovým a desktopovým klientem. Byly zmíněny některé specifické problémy spojené s vývojem softwaru pro eye tracking, které byly v průběhu vytváření aplikace nutné řešit.

V rámci práce byla vytvořena samostatná knihovna v jazyce Java, která slouží především pro generování vizualizací heatmap a gaze plot. Pro generování těchto vizualizací byly navrženy a implementovány vlastní algoritmy. Byla orientačně otestována i rychlost těchto algoritmů, především pro ověření toho, jaký vliv na rychlost výpočtu má počet využitých vláken při generování heat map.

9 Seznam použité literatury

[1] *Apache HttpComponents* [online]. 2016 [cit. 2016-02-18]. Dostupné z: https://hc.apache.org/

[2] BAUER, Christian a Gavin KING. *Java persistence with Hibernate*. Greenwich: Manning, 2007. ISBN 978-193-2394-887.

[3] BERGSTEN, Hans. *JavaServer pages*. 2. vyd. Sebastopol, California: O'Reilly, 2002. ISBN 05-960-0317-X.

[4] BERGSTROM, Jennifer a Andrew SCHALL. *Eye Tracking in User Experience Design*. USA: Elsevier, 2014. ISBN 978-0-12-408-138-3.

[5] BOJKO, Aga a Robert SCHUMACHER. *Eye Tracking and Usability Testing in Form Layout Evaluation* [online]. 2008 [cit. 2016-01-30]. Dostupné z: http://c.ymcdn.com/sites/www.bfma.org/resource/resmgr/Articles/08_32.pdf

[6] BOJKO, Aga. *Eye Tracking the User Experience: A practical Guide to Research*. New York: Rosenfeld Media Brooklyn, 2013. ISBN 1-933820-10-1.

[7] CASEY, John, Vincent MASSOL, Brett PORTER, Carlos SANCHEZ a Jason VAN ZYL. *Better Builds With Maven* [online]. 2006 [cit. 2016-02-16]. Dostupné z: http://www.scribd.com/doc/238927/Better-Builds-With-Maven

[8] DUCHOWSKI, Andrew. *Eye Tracking Methodology: Theory and Practice*. Second Edition. Clemson: Springer, 2007. ISBN 978-1-84628-608-7.

[9] EBBERS, Hendrik a Michael HEINRICHS. JavaFX 8. *DZone: Refcardz* [online]. 2015 [cit. 2016-02-17]. Dostupné z: https://dzone.com/refcardz/javafx-8-1

[10] EVANS, Karen, Robert JACOBS, John TARDUNO a Jeff PELZ. Collecting and Analyzing Eye-tracking Data in Outdoor Environments. *Journal of Eye Movement Research* [online]. 2012 [cit. 2016-01-18]. Dostupné z: http://www.bcs.rochester.edu/people/robbie/evans-etal_jemr2012.pdf

[11] *EyeProof | Analytics* [online]. 2014 [cit. 2016-01-30]. Dostupné z: http://www.eyeproof.net/

[12] *Hibernate Developer Guide* [online]. 2015 [cit. 2016-02-16]. Dostupné z: http://docs.jboss.org/hibernate/orm/4.3/devguide/en-US/html_single/

[13] Java Heatmap (Example Code). *Software Talk* [online]. 2012 [cit. 2016-02-02]. Dostupné z: http://software-talk.org/blog/2012/01/java-heatmap-example/

[14] Java: Getting Frame Data. *The Eye Tribe: Developer site* [online]. 2014 [cit. 2016-02-13]. Dostupné z: http://dev.theeyetribe.com/java/

[15] KLIMEŠ, Jeroným. Způsoby sledování pohybu zraku. *Strategie: Oční kamera jako bič na kreativce* [online]. 2001 [cit. 2016-01-22]. Dostupné z: http://klimes.mysteria.cz/clanky/psychologie/ocnikamera_historie.pdf

[16] LEGGETT, David. A Brief History of Eye-Tracking. *UX Booth* [online]. 2010 [cit. 2016-01-11]. Dostupné z: http://www.uxbooth.com/articles/a-brief-history-of-eye-tracking/

[17] LUPU, ROBERT GABRIEL a FLORINA UNGUREANU. *A survey of eye tracking methods and applications* [online]. 2013 [cit. 2016-01-10]. Dostupné z: http://www12.tuiasi.ro/users/103/071-086_006_Lupu_.pdf

[18] *Products – The Eye Tribe* [online]. 2016 [cit. 2016-01-14]. Dostupné z: https://theeyetribe.com/products/

[19] *Spring Data JPA - Reference Documentation* [online]. 2015 [cit. 2016-02-16]. Dostupné z: http://docs.spring.io/spring-data/jpa/docs/1.9.2.RELEASE/reference/html/

[20] *Spring Framework Reference Documentation* [online]. 2015 [cit. 2016-02-13]. Dostupné z: http://docs.spring.io/spring/docs/4.2.4.RELEASE/spring-framework-reference/htmlsingle/

[21] *Spring Roo - Reference Documentation* [online]. 2014 [cit. 2016-02-16]. Dostupné z: http://docs.spring.io/spring-roo/docs/1.3.2.RELEASE/reference/html/

[22] The Eye Tribe Java SDK. *GitHub* [online]. 2016 [cit. 2016-02-12]. Dostupné z: https://github.com/EyeTribe/tet-java-client

[23] *The Eye Tribe: Developer site* [online]. 2014 [cit. 2016-01-12]. Dostupné z: http://dev.theeyetribe.com/

[24] TSAGKLIS, Ilias. Introduction to Xuggler for Video Manipulation. *Java Code Geeks* [online]. 2011 [cit. 2016-02-17]. Dostupné z: http://www.javacodegeeks.com/2011/02/introduction-xuggler-video-manipulation.html

[25] Webcam Capture API. *GitHub* [online]. 2015 [cit. 2016-02-18]. Dostupné z: https://github.com/sarxos/webcam-capture

10 Přílohy

1) Příloha 1 – Struktura přiloženého CD

Přiložené CD má následující adresářovou strukturu:

- **aplikace**
  - **client**
    - **runnable_jar**
      - **The_EyeTribe_App.jar** – *spustitelná verze desktopového klienta*
      - **app.properties** – *soubor, ve kterém lze nastavit adresu serveru*
      - **lib** – *složka s potřebnými knihovnami*
    - **src** – *složka se zdrojovými kódy desktopového klienta*
  - **server**
    - **The_EyeTribe_App.war** - *serverová část aplikace (WAR)*
    - **src –** *složka se zdrojovými kódy serverové části*
  - **heatmap-library**
    - **heatmap-library.jar** – *knihovna pro generování vizualizací*
    - **src –** *složka se zdrojovými kódy knihovny*
  - **README.txt** – *soubor s pokyny pro zprovoznění aplikace*
- **text_prace**
  - **novotny_jan_diplomova_prace.pdf** – *text práce ve formátu PDF*

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2015/2016

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai2-p)

Podklad pro zadání DIPLOMOVÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Novotný Jan	U Sopky 1887, Nová Paka	I1472

TÉMA ČESKY:

Eye tracking pomocí kamery The Eye Tribe

TÉMA ANGLICKY:

Eye Tracking Using The Eye Tribe Tracker

VEDOUCÍ PRÁCE:

doc. Mgr. Tomáš Kozel, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl práce: Seznámit se s problematikou uživatelského testování prostřednictvím oční kamery a navrhnout a implementovat software k vyhodnocování dat z uživatelského testování s oční kamerou.

Osnova práce:
1) Úvod
2) Obecná problematika Eye tracking
3) Popis využitelných technologií
4) Návrh a implementace softwaru
5) Výsledky
6) Závěr

SEZNAM DOPORUČENÉ LITERATURY:

--

Podpis studenta: ..... **Datum:** 8.10.2015

Podpis vedoucího práce: ..... **Datum:** 8.10.2015