VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

INTELIGENTNÍ VIDEO-PŘEHRÁVAČ

INTELIGENT VIDEO-PLAYER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **DAVID SABELA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MICHAL KAPINUS**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19416/2016/xsabel01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Sabela David**
Obor: Informační technologie
Téma: **Inteligentní video-přehrávač**
Inteligent Video-Player
Kategorie: Uživatelská rozhraní

Pokyny:

1. Seznamte se s existujícími aplikacemi pro efektivní navigaci a přehrávání multimediálních dat z dohledových systémů. Prostudujte moderní přístupy návrhu a realizace uživatelských rozhraní pomocí webových technologií.
2. Navrhněte uživatelské rozhraní inteligentního video-přehrávače, včetně potřebných datových struktur popisujících zájmové části videa.
3. Implementujte klíčové moduly a funkční základ systému s využitím existujících nástrojů a knihoven. Dbejte na technickou kvalitu řešení, modularitu a efektivní udržitelnost.
4. Naplňte systém vhodnými daty a demonstrujte funkčnost řešení. Proveďte experimenty na uživatelskou použitelnost a technickou efektivitu řešení. Komentujte výsledky experimentů a diskutujte možnosti dalšího vývoje.
5. Vytvořte plakát nebo krátké video prezentující klíčové výsledky vašeho řešení.

Literatura:

- M. Sonka, V. Hlaváč, R. Boyle. *Image Processing, Analysis, and Machine Vision, CL-Engineering*, ISBN-13: 978-0495082521, 2007.
- Dále dle pokynu vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2, 3 a částečně bod 4.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kapinus Michal, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cílem práce bylo vytvořit uživatelské rozhraní pro zobrazování zajímavých údálostí z dohledových kamerových systémů. V části teorie jsou popsány technologie, které jsou vhodné k vytvoření webové aplikace, která pracuje s videem. Jsou zde popsány také zásady kvalitního grafického uživatelského rozhraní a je zde příblížena problematika dohledových kamerových systémů. V části návrh a implementace je popsáno mé řešení prototypu inteligentního video-přehrávače. Výsledný prototyp je otestován řadou testujících a výsledky jsou analyzovány a vyhodnoceny v části testování.

Abstract

The aim of this bachelor thesis was to create a user interface for displaying interesting or in different way significant events from the views of surveillance camera systems. In the first part of the thesis - Theory - technologies useful for the creation of a web application working with video are described. Principles of graphical quality in user interferences are also summarized in this part, as well as explanation of the principal questions of the surveillance camera systems problematics. In the second and third part - Design and Implementation - my own solution of the prototype of inteligent video player is depicted. The resulting prototype is thereafter subjected to examination by several testers whose results are analysed and evaluated in the final part - Testing.

Klíčová slova

HTML, CSS, JavaScript, jQuery, grafické uživatelské rozhraní, dohledové kamerové systémy, responzivní design, hacky, filtry, ECMAScript, DOM, BOM, canvas, video, Evidant, časová osa.

Keywords

HTML, CSS, JavaScript, jQuery, graphical user interface, surveillance systems, responsive design, hacks, filters, ECMAScript, DOM, BOM, canvas, video, Evidant, timeline.

Citace

SABELA, David. *Inteligentní video-přehrávač*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Kapinus Michal.

Inteligentní video-přehrávač

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Kapinuse. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
David Sabela
15. května 2017

Poděkování

Děkuji panu Ing. Michalu Kapinusovi za odborné vedení, trpělivost a ochotu, kterou mi v průběhu bakalářské práce věnoval.

Obsah

Kapitola 1

Úvod

Internet propojuje miliony počítačů po celém světě, poskytuje globální komunikaci, úložný prostor, výpočetní infrastrukturu a je integrován v mobilní i bezdrátové technologii. Počet zařízení využívající internet stále roste a připojení k internetu je stále rychlejší a rychlejší.

V současné době roste potřeba inteligentních vizuálních sledovacích systémů v komerčních, policejních a vojenských aplikacích. Schopnost rozpoznávat objekty, lidi a popisovat jejich činnosti je nezbytné pro automatizovaný vizuální dohled.

Cílem mé práce bylo vytvoření webového inteligentního video-přehrávače. K dosažení tohoto cíle byly použity jazyky JavaScript, HTML[1], CSS[2] a framework jQuery.

Tato práce je rozdělena do čtyř hlavních částí. První část je věnována teorii, která je nutná k zvládnutí výše uvedeného cíle. V další části návrh, je popsáno již existující řešení a návrh mého nového řešení. Následuje část implementace, zde je popsán způsob, jakým byl vytovořen prototyp mého řešení a část testování, kde jsou popsány metody, které byli použity, analýza a vyhodnocení výsledků těchto metod.

[1] HTML - Hypertext Markup Language
[2] CSS - Cascading styles Sheets

Kapitola 2

Teorie

Tato kapitola je zaměřena převážně na technologie k vytvoření webové stránky či aplikace, která umí pracovat s videem. Kapitola popisuje tři jazyky, které byly použity k realizaci inteligentního video-přehrávače.

Při vývoji webové stránky (aplikace) je dobré stránku vyvíjet po vrstvách a odděleně. Když se vytváří webová stránka postupuje se z dola nahoru. Nejdříve se napíše obsah v HTML, to je základní vrstva, která se zobrazí všem navštěvníkům webu, bez ohledu na to, jaký prohlížeč používají. Po vytvoření základní vrstvy nastupuje druhá vrstva, a to prezentační, zde se kóduje design stránky v jazyce CSS. Poslední vrstvou je chování. Tato vrstva zavádí dynamické chování a interaktivitu do webové stránky. Při vývoji je dobré držet se kázně a tyto vrstvy mít oddělené [18].

2.1 HTML

HTML je základní značkovací jazyk určený k tvorbě webových stránek a je dominantním jazykem na webu. Vznikl proto, aby bylo možné formátovat a sdílet vědecké materiály. V současné době jej používají téměř všechny typy dokumentů. Dnes je již HTML verze 5.1, která obsahuje nové prvky, ale také všechny platné prvky předchozích verzí, dále zahrnuje nové funkce, vylepšení stávajících funkcí a různá API[1] založená na skriptech. Tato kapitola je zaměřena převážně na využití prvku video v jazyce HTML [11, 15, 13, 5].

Historie

První definice jazyka HTML je datována na rok 1991 a vytvořil ji Tim Berners-Lee, jako součást projektu WWW[2]. Tato verze je známa jako HTML 0.9 a obsahovala možnost zvýraznění textu, přídání obrázků či odkazů, a také rozčlenění do logických úrovní. Časem již dosavadní HTML nestačilo na rostoucí požadavky a bylo obohaceno o nové prvky. Vznikla verze HTML 2.0, která rozšiřuje dosavadní verzi a definuje práci s formuláři. Další rozšíření, které je známé jako HTML 3.0 zahrnuje vytváření tabulek, matematických vzorců a stylování objektů. Následovali verze 3.2 (1997), 4.0 (1997), 4.01 (1999), HTML 5 (2014) a HTML 5.1 (2016) [13, 5].

[1] API - Appliction Programing Interface, jedná se o rozhraní určené pro kód.
[2] WWW - World Wide Web - celosvětová síť.

Struktura dokumentu

Pro správnou kostru dokumentu je potřeba použít následující elementy viz kód 2.1. První řádek určuje použitou verzi jazyka HTML. Element `<html>` je jakýmsi celkovým kontejnerem dokumentu a obsahuje hlavičku `<head>` a tělo `<body>`. Hlavička může obsahovat nepovinné tagy (title, meta, link, style, script aj.). Pokud se vyskytuje v hlavičce prostý text, je možné, že se v některých prohlížečích zobrazí na začátku stránky. Element `<body>` obsahuje veškerý zobrazovaný obsah stránky.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
  <title> document title </title>
</head>
<body>
   <p>document content</p>
</body>
</html>
```

Kód 2.1: Šablona HTML dokumentu (převzato z[5]).

Video

Video je možné do webové stránky vložit pomocí prvku `<video>`, ovšem ne všechny prohlížeče jej dokáží spustit ve všech formátech. V tabulce 2.1 je možné vidět, které kodeky jsou podporované a u jakých prohlížečů.

	WebM	Ogg Theora	MPEG-4 H.264
Firefox	✓	✓	✗
Safari	✗	✗	✓
Opera	✓	✓	✗
Google Chrome	✓	✓	✓
Internet Explorer	✗	✗	✓

Tabulka 2.1: Podporované kodeky v jednotlivých prohlížečích (převzato z[11]).

Atributy pro video element

- src - odkaz na zdroj videa
- autoplay - automatické přehrávání po načtení videa
- loop - opakované spuštění
- poster - odkaz na obrázek, který reprezentuje video
- preload - jak velká část videa se má načíst ze zdroje
- width a height - změna šířky a výšky
- controls - výchozí ovládací prvky

Canvas

Prvek canvas se používá v jazyce HTML pro kreslení grafiky pomocí JavaScriptu. Díky němu je možné vytvářet grafy, fotky, jednoduché animace a dokonce je možné manipulovat s pixely z obrázků či videa. Prvek canvas má pouze dva atributy a to *height* (výška) a *width* (šířka).

2.2 Kaskádové styly - CSS

Kaskádové stylové předpisy, anglicky Cascading styles Sheets, jsou skvělým pomocníkem pro vytváření vzhledu dokumentu nebo kolekcí dokumentů. K použití CSS je nutné mít dokument, bez něj jsou kaskádové styly v podstatě k ničemu. Kaskádové styly vznikly kvůli rychle narůstajícímu počtu webových stránek a obrovskému tlaku na vylepšování vzhledu webových stránek.

Na počátku webu na vytváření webových stránek jazyk HTML stačil, skládal se ze strukturálních prvků, jimiž se popisovaly odstavce, seznamy atd. HTML se nesoustředilo na vzhled, bylo to jen jednoduché značkovací schéma. Vše se změnilo po příchodu prvního moderního webového prohlížeče Mosaicu. Začal obrovský nárust webových stránek korporací, osobních deníků, stránek universit a řady dalších druhů webových stránek. S nárustem počtu webů rostly i požadavky na nové prvky a funkce. Důsledkem tlaku se začaly do jazyka vkládat nové prvky jako `<FONT>` a `<BIG>` a z jazyka HTML se začal tvořit prezentační jazyk. Tímto namátkovým přidáváním prvků časem vznikly problémy. Webové stránky totiž obsahovaly obrovské počty značek a navíc u většiny případů to byly prvky font, které nemají žádný sémantický význam ohledně toho, co prezentují viz kód 2.2, zde je možné vidět, že pomocí prvku font prohlížeči není řečeno nic ohledně toho, co popisuje. Správně by to mělo být napsáno pomocí značky `<h1>` nebo obdobné, která má význam nadpisu.

```
<font size="3" face="Hevetica" color="red" > Title </font>
```

Kód 2.2: Ukázka použití prvku font.

Autoři webových stránek se vzdávali kontroly kvůli designu, a to přineslo řadu problémů. Nestrukturované stránky ztěžují hledání informací. Výkonné vyhledávací enginy hledají informace v nadpisech, v sekci záhlaví, v textu odstavců, aby toto bylo možné realizovat, musí být stránky správně označkované. Pokud bude webová stránka dobře strukturovaná zlepší to pozici ve výsledcích vyhledávání. Webové stránky, které nejsou strukturované jsou velice špatně udržovatelné.

Toto vše vedlo k hledání řešení a W3C[3] v roce 1995 zveřejnilo CSS a v roce 1996 se tento materiál dostal do stavu Recomendation se stejnou vahou jako samotný jazyk HTML [2, 17, 9, 10].

Způsoby vložení CSS

Existují tři způsoby pro vložení kaskádových stylů do HTML dokumentu. Tyto tři způsoby budou popsány v následujících podkapitolách [2].

[3] W3C - World Wide Web Consortium

Značka Link

Jedním ze způsobů, jak připojit CSS soubor k HTML, je pomocí značky link. Značka link se vkládá do hlavičky v HTML dokumentu a obsahuje atributy *rel* (vztah), *type* (typ dat, které se načtou pomocí značky link), *href* (obsahuje url souboru) a *media* (aplikace stylu do specifikovaných prezentačních medií).

```
<link rel="stylesheet" type="text/css" href="soubor.css" media="all" />
```

Kód 2.3: Připojení externího CSS souboru k dokumentu HTML.

Značka style

Dalším způsobem jak vložit kaskádové styly do HTML dokumentu je pomocí značky style. Značka style vždy musí začínat `<style type="text/css">` a končit `</style>`. Pomocí style se CSS předpisy piší rovnou do HTML dokumentu a bývají označovány za vložené stylové předpisy. Do kontejneru `<style>` se mohou připojovat i externí CSS soubory, a to přes direktivu `@import`. Těchto direktiv může být importováno více než pouze jedna, ale je nutné, aby se `@import` direktiva nacházela před ostatními předpisy viz kód 2.4.

```
<style type="text/css">
@import url(soubor.css);
p {color : green;}
 </style>
```

Kód 2.4: Ukázka použití prvku style v dokumentu HTML (převzato z[2]).

Inline styly

Styly je možné vepisovat přímo ke konkrétnímu prvku, tomuto případu se říká inline styly.

```
<p style="color:green" > paragraph </p>
```

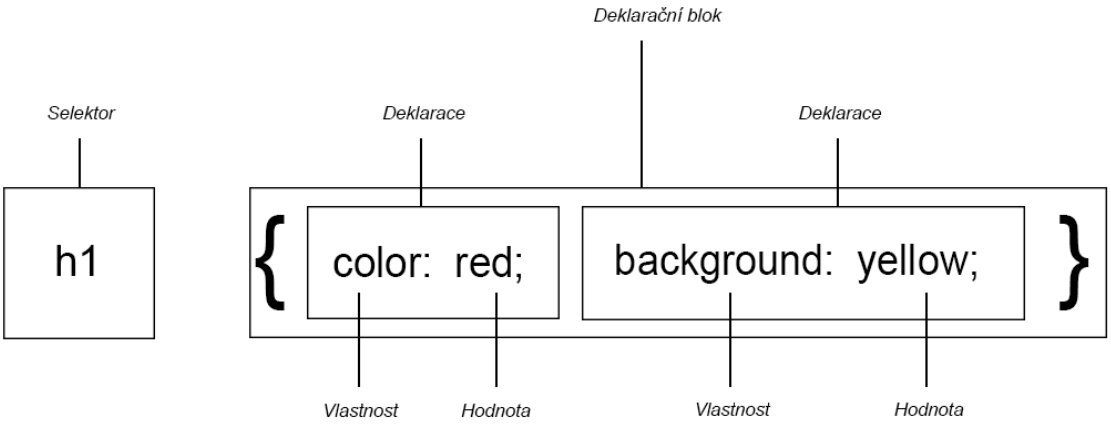
Kód 2.5: Ukázka inline stylu (převzato z[2]).

Syntaxe

Pravidlo CSS se skádá ze selektoru a deklaračního bloku. Nejčastěji bývá selektorem značka html, ale nemusí to tak být vždy (např. XML prvek). Deklarační blok obsahuje dvojici vlastnost a hodnotu oddělenou dvojtečkou. Pravidlo může obsahovat více těchto dvojic. Selektorů může být pro dané pravidlo také více viz obr. 2.1.

Selektory

Jak už bylo zmíněno selektory jsou nejčastěji prvky html, ale existují ještě další typy, a to selektory ID, selektory tříd a univerzální selektor. Univerzální selektor je značen znakem * a reprezentuje všechny prvky v dokumentu. Selektor ID by měl být v dokumentu jednoznačně identifikovatelný, tudíž jeho použití by mělo být pouze pro jeden konkrétní případ. Ve skutečnosti nedochází k žádné kontrole, ale mohou pak nastat problémy např. při použití funkce JavaScriptu `document.getElementById()`, která je závislá na jedinečnosti ID. Přiřazování pravidel k ID je trošku jiné, než k normálním značkám html, zde se před ID píše znak `#`.



Obrázek 2.1: Syntexe kaskádových stylů (převzato z [2])

Dalším selektorem je třída. Hlavní rozdíl mezi ID a třídou je, že třídu můžeme bez problému aplikovat na více prvků v dokumentu. Pravidla pro třídu se zapisují tak, že před název třídy vložíme znak tečka. Třídy mohou obsahovat více slov (názvů), v takovém případě jsou na prvek aplikovány pravidla obou názvů třídy a je jedno v jakém pořadí jsou napsány.

V situaci, kdy je nějaké pravidlo stejné pro více prvků, je vhodné použít seskupování selektorů. V programování je nepsané pravidlo neopakuj stejný kód vícekrát, tudíž i v CSS je možné toto pravidlo naplnit, namísto `h1 { color:red; } h2 { color:red; }` je možné napsat `h1, h2 { color:red; }`. Seskupovat lze i deklarace a to následovně `h1, h2 { color:red; font-size:12px; }`.

CSS využívá strukturu dokumentu HTML pro určování a aplikaci stylů. V této hierarchii má každý prvek pevné místo. Každý prvek v dokumentu je buď rodičem, nebo potomkem jiného prvku a často obojím zároveň. Pojmy rodič a potomek znamenají v této hierarchii vztah dvou prvků, kdy jeden prvek je přesně jednu úroveň pod druhým. Pokud jsou mezi prvky dvě a více úrovní, jedná se o vztah předchůdce a následník.

Specifičnost a dědičnost

V případě, že má prvek více pravidel je v CSS takzvaná specifičnost selektoru. Hodnota specifičnosti je určena čtyřmi čísly ve formátu 0,0,0,0 a určuje se následovně:

- hodnota atributu ID = 0,1,0,0,
- hodnota atributu class = 0,0,1,0,
- hodnota pro každý prvek = 0,0,0,1,
- hodnota pro inline styly = 1,0,0,0.

Podle těchto hodnot jsou pak upřednostňována ta pravidla, která jsou obsažená v selektoru s vyšší hodnotou.

Dědičnost způsobuje, že se pravidla neaplikují pouze na daný prvek, ale také na jeho následníky. Je důležité vědět, že se nedědí všechny vlastnosti např. okraje, výplň a orámování se nedědí. Kvůli různé implementaci prohlížečů není možné počítat s tím, že dědičnost bude fungovat všude stejně.

Responzivní design

Pro vytvoření responzivního designu je možné použít řadu frameworků např. Bootstrap, Materialize atd. Tato podkapitola se zaměřuje na vytvoření responzivního gridu bez použití frameworků. Je důležité, aby prohlížeč věděl, že při změně velikosti stránky nemá obsah nijak měnit viz 2.6 [7].

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" >
```

Kód 2.6: Prohlížeč při zmenšení okna nebude měnit velikost obsahu.

Poté je možné nadefinovat výchozí design např. pro dnes již standardní Full HD rozlišení. Pokud má být stránka responzivní v závisloti na změně velikosti okna, pak se použije příkaz `@media`, který se skládá ze dvou částí: první část je media typ a bezprostředně poté je samotný dotaz, který je rozdělen zase do dvou částí, a to na název vlastnosti a hodnotu viz kód 2.7.

```
@media screen and (min-width: 1024px) {
  body {
          font-size: 100%;
      }
}
```

Kód 2.7: Styl se aplikuje jakmile bude okno větší než 1024px (převzato z[2]).

Hacky a filtry

V dnešní době i dobře napsaný CSS kód nezaručuje, že styly budou fungovat na všech prohlížečích. Prohlížeče obsahují nedostatky, a aby vývojáři mohli i tak vytvářet kvalitní webové stránky, museli začít využívat nestandardní postupy (hacky, filtry).

Hacky a filtry jsou mocnou zbraní webových vývojářů. CSS hacky (patche) se využívají k obcházení chyb webových prohlížečů tím, že využívají filtry a pomocí nich vyfiltrují pravidla, která pro daný prohlížeč nemají být použita a pravidla, která budou použita. Pomocí CSS hacků jde i nekorektní implementací dosáhnout k tíženému chování prohlížeče. Dá se říci, že CSS filtry jsou specifický typ hacku. Použití hacků se doporučuje až jako poslední možnost, protože se tím riskuje nebezpečí, že některý prohlížeč takový zápis nebude umět zpracovat. CSS hacků je celá řada, zde je pouze pár vybraných [2].

Podmíněné komentáře

Jedná se o nestandardní rozšíření Microsoftu běžných HTML komentářů, pro ostatní prohlížeče se jeví jako normální komentář, ovšem pro Internet Explorer nikoliv. Pomocí podmíněných komentářů je možné pro konkrétní verze IE[4] určit speciální styl. Příklad použití viz kód 2.8.

```
<!-- [if IE]
 <style type="text/css">
@import ("spec.css");
</style>
-->
```

Kód 2.8: Pouze Internet Explorer importuje spec.css (převzato z[2]).

[4]Internet Explorer

Hvězdičkový hack

Hvězdičkový hack je jeden s nejznámějších a také je velmi jednoduchý na zapamatování. Jeho funkce spočívá v tom, že univerzální selektor se vloží před běžný typový selektor html a tím ukryje toto pravidlo pro všechny webové prohlížeče, až na Internet Explorer, toto se používá v situacích, kdy vývojář chce použít nějaká pravidla pouze pro Internet Explorer.

```
* html {
  font-size: small;
}
```

Kód 2.9: Hvězdičkový hack (převzato z[2]).

Podtržítkový hack

Tento hack využívá změnu názvu v deklaraci podržítkem a tím docílí, že nové prohlížeče toto pravidlo ignorují ovšem IE starších verzí nikoliv a pravidlo použije.

```
#nav {
position: fixed;
_position: static;
}
```

Kód 2.10: Potržítkový hack (převzato z[2]).

2.3 JavaScript

Jde o velmi rychlý a populární skriptovací jazyk. Jeho rychlost a relativní jednoduchost ho učinily nejoblíbenějším jazykem na webu. Vznik tohoto jazyka byl oznámen 4. prosince roku 1995 na společné konferenci společností Netscapea Sun. JavaScript je ochranou známkou právě společnosti Sun Miscrosystems. Důvod vzniku JavaScriptu bylo omezit komunikaci se servery, která výrazně vše prodlužovala, především validace formulářů. Díky velké popularitě JavaScriptu se objevovaly jeho různé varianty od jiných firem např. od Microsoftu a to zdůraznilo fakt, že JavaScript nemá žádné standardy určující jeho syntaxy nebo rysy. V roce 1997 byl organizací ECMA[5] předán JavaScript 1.1 jako návrh pro standardizaci. Po několika měsících tak vznikl standard ECMA-262 s názvem ECMAScript. V následujících letech jej přijaly také organizace ISO/IEC[6] jako standard ISO/IEC-16262. Existuje spousta důvodů proč je JavaScript stále tak populární a další jsou neustále vynalézány. JavaScript oživuje stránky, umožňuje ověřování vstupních dat, má CGI[7] na straně klienta, ulehčuje zaneprázdněným serverům (řadu úkolů zpracuje přímo v prohlížeči) a zpracovává cookies. JavaScript může být aplikován řadou způsobů a je využíván na stránkách po celém internetu. JavaScript je plnohodnotným programovacím jazykem, který je schopný provádět složité výpočty, interakce a dokonce i metaprogramování. Tvoří jej tři odlišné části, a to jádro (ECMAScript), objektový model dokumentu(DOM[8]) a objektový model prohlížeče(BOM[9]).

[5]ECMA - European Computer Manufacturers Association(sdružení evropských výrobců počítačů).

[6]International Organization for Standartization/ International Electrotechnical Comission - Mezinárodní organizace pro standardizaci/ Mezinárodní elektrotechnický výbor.

[7]Common Gateway Interface - Základní výměnné prostředí pro příjem vstupu a publikování výstupu jakéhokoliv programu (třeba z Pascalu nebo Perlu)

[8]Document Object Model

[9]Browser Object Model

ECMAScript

ECMAScript je popis jazyka implementujícího všechny aspekty popsané ve specifikaci, není vázán s webovými prohlížeči, neobsahuje metody pro vstup či výstup. ECMAScript obsahuje na základní úrovni tyto části jazyka:

- syntaxe
- typy
- příkazy
- klíčová slova
- vyhrazená slova
- operátory
- objekty

Různé verze ECMAScriptu jsou definovány jako edice. Momentálně je standardní verze ES[10] 5, která je podporována ve všech prohlížečích. Existuje však i ES 6 a ES 7. ECMAScript verze 6 je výrazný posun vpřed, obsahuje totiž třídy a moduly, které jsou definovány sémanticky stejně jako v ES 5, ale také iterátory, generátory vyjímek a kolekce. K použití ES 6 je nutné mít překladač např. Babel[11], který přeloží kód z ES 6 do ES 5, aby ho webový prohlížeč dokázal spustit. Některé z novinek viz kód 2.11 a 2.12 další je možné najít viz[12].

```
// ES 6
var list = [1,2,3]
var [a,,b] = list
[b,a] = [a,b]
}

// ES 5
var list = [1,2,3];
var a = list[0], b = list[2];
var tmp = a; a = b; b = tmp;
```

Kód 2.11: Decentralizační přiřazení (převzato z[13]).

```
// ES 6
var customer = { name: "Foo"}
var card = { amount: 7, product: "Bar", unitprice: 42 }
var message = `Hello ${customer.name}, want to buy ${card.amount} ${card.
    product} for a total of ${card.amount * card.unitprice} bucks?`

// ES 5
var customer = { name: "Foo"};
var card = { amount: 7, product: "Bar", unitprice: 42 };
var message = "Hello " + customer.name +",\n" + "want to buy "+ card.amount
    +" " + card.product + " for\n" + "a total of " +(card.amount * card.
    unitprice) +" bucks?";
```

Kód 2.12: Interpolace řetězce (převzato z[14]).

[10] ES - ECMAScript
[11] https://babeljs.io/
[12] http://es6-features.org/
[13] http://es6-features.org/ArrayMatching
[14] http://es6-features.org/StringInterpolation

Objektový model dokumentu - DOM

Objektový model dokumentu je aplikační rozhraní pro jazyk XHTML[15] a HTML. DOM mapuje celou webovou stránku do stromové hierarchie uzlů. Pomocí tohoto rozhraní je možné pracovat se samotnou webovou stránkou. DOM umožňuje přistupovat k jednotlivým prvkům dokumentu a díky stromové struktuře je možné rozlišovat nadřazené, podřazené nebo rovnocenné elementy. Pomocí aplikačního rozhraní modelu DOM je možné uzly odstraňovat, přidávat, nahrazovat a upravovat.

Objektový model prohlížeče - BOM

Objektový model prohlížeče umožňuje komunikaci s prohlížečem mimo kontext zobrazované stránky. BOM pracuje s oknem prohlížeče a je jedinou částí JavaScriptu, která nemá žádný standard. V hierarchii BOM je window tzv. top-level objekt. Navazují na něj tyto objekty:

- navigator - reprezuntuje informace o prohlížeči a operačním systému
- screen - obsahuje informace o obrazovce počítače klienta
- history - obsahuje informace o navštívených stránkách
- location - obsahuje informace o aktuálním URL
- document - reprezentuje aktuální webovou stránku

Připojení JavaScriptu do HTML

Připojit JavaScript do HTML dokumentu je možné interním, externím a In-line zápisem. Pro interní zápis stačí použít párový tag `<script>` a do něj napsat JavaScriptový kód. V případě externího zápisu se používá také párový tag `<script>`, ale bez těla, pouze se do atributu `src` napíše cesta k souboru. In-line zápis nepoužívá tag `<script>` vepisuje se přímo jako atribut do jiného tagu.

JavaScript a canvas

Element canvas byl již popsán v podkapitole 2.1. Zde bude popsáno jak canvas použít vrámci práce s videem.

```
var canvas = document.getElementById("mycanvas");
var ctx = canvas.getContext("2d");
ctx.rect(10,10,100,100);
ctx.fill();
```

Kód 2.13: Vykreslení čtverce na plátno (převzato z [11]).

V první řadě je nutné přiřadit prvek do proměnné, poté pomocí funkce `getContext("2d")` je možné měnit obsah plátna. Tento postup je nezbytný pro další práci s plátnem. Funkce `rect` vykreslí na plátno čtverec a funkce `fill` jej vyplní barvou. Pomocí funkce `getImageData` je možné získat informace o pixelech v daném rozmezí. Podobným způsobem se postupuje v případě vykreslí snímku z videa viz kód 2.14.

[15] XHTML - extensible hypertext markup language (rozšiřitelný hypertextový značkovací jazyk)

```
 var canvas = document.getElementById("mycanvas");
 var video = document.getElementById("myvideo");
 var ctx = canvas.getContext("2d");
 canvas.width = 190;
 canvas.height = 80;

 if(video.paused){
  ctx.drawImage(video,0,0,190,80);
 }
}
```

Kód 2.14: Vykreslení snímku videa na plátno (převzato z [11]).

V tomto příkladě je navíc oproti předchozímu element video, ze kterého je překreslen aktuální snímek v době pauzy a ten pak je zakreslen na plátno pomocí funkce `drawImage`.

2.4 Grafické uživatelské rozhraní

Uživatelské rozhraní je soubor postupů týkajících se užívání daného objektu. Grafické uživatelské rozhraní je rozhraní, kdy objekty jsou reprezentovány graficky a uživateli je umožněno s nimi manipulovat pomocí ukazovacího zařízení. Ovládání je realizováno pomocí formulářů s ovládacími prvky, menu a také přímou manipulací. Přímá manipulace je dnes již samozřejmostí. Uživatel vidí průběh akcí, může tyto akce vrátit zpět, objekty jsou zobrazovány jako metafory reálného světa, to vše je tu z důvodu usnadnění práce a zlepšení orientace.

Aplikace může být sebelépe naprogramovaná, ale bez kvalitního uživatelského rozhraní nebude dobře použitelná.

Již v šedesátých letech dvacáteho století byly snahy o grafické uživatelské rozhraní. Významný projekt Xerox PARC, kde bylo definováno tzv. WIMP[16] paradigma. Další významný posun byl počítač Lisa od Apple Computer, který měl pracovní plochu s ikonami a rozbalovací menu.

Nyní moderní grafická uživatelská rozhraní směřují k adaptivnosti, tedy co možná nejvíce se přizpůsobit uživatelům a jejich potřebám. Hlavním cílem při vytváření uživatelského rozhraní je, aby práce s programem byla snadná, produktivní a příjemná. Správné rozhraní by mělo fungovat jako prodloužení člověka, to znamená, že software by měl reflektovat schopnosti uživatele a odrážet je v jeho potřebách. Grafické uživatelské rozhraní musí být intuitivní nebo snadno pochopitelné. Při vývoji je nutné myslet na koncového zákazníka, protože každý člověk nemá stejné zkušenosti s počítači [12, 16, 4].

Design

Při návrhu designu je důležité, aby dobrý vzhled nebyl na úkor funkčnosti. I zde platí pravidlo méně je někdy více. Důležitou součástí designu jsou barvy. Barvy jsou užitečným pomocníkem pro zvýrazňování důležitých prvků. Je vhodné volit takové barvy, aby splňovaly zažitý význam u uživatelů, například výrazné barvy se hodí na upozorňování, naproti tomu nevýrazné barvy není vhodné používat pro výstrahu. U správného designu hraje neméně významnou roli rozložení prvků. Prvky, které obsahují relevantní informace by měly být pospolu. Samotné rozložení prvků záleží na vnímání uživatele, kam soustředí pohled při hledání informací. Jedním ze základních faktů je, že při hledání uživatel směřuje pohled od

[16] Windows, Icons, Menus, Pointing device

levého horního rohu a dále po směru hodinových ručiček, tudíž je vhodné i tak rozmisťovat důležité prvky.

Schneidermanova pravidla pro design prostředí

- Usilování o konzistenci v sekvencích akcí, rozvržení obrazovky, terminologii apod.
- Umožnění častým uživatelům používat zkratky.
- Nabídnutí informativních zpětných vazeb podle důležitosti akce, o níž informuje.
- Při vykonání příkazu zajistit, aby o tom uživatel věděl.
- Předcházení chyb a nabízení jednoduchých řešení, pokud chyba nastane.
- Pro provádění experimentů se systémem zajistit vracení akcí.
- Vytvoření ovládacích prvků pro manipulaci se systémem.
- Šetření krátkodobé paměti uživatele designově jednoduchých obrazovek [12].

2.5 Dohledové kamerové systémy

Systémy pro vizuální dohled jsou vyžadovány čím dál více a to především z důvodu zajištění vyšší bezpečnosti. Kamerové systémy pro vizuální dohled musí pracovat nepřetržitě a s co nejmenším počtem výpadků, musí zpracovávat velké množství dat v reálném čase a reagovat na ně. Inteligentní dohledové systémy jsou jedním z nejaktivnějších výzkumů v oblasti počítačového vidění. Cílem těchto systémů je efektivním způsobem vytáhnout vše podstatné z rozsáhlých záznamů z video kamer. Sledování má širokou škálu aplikování například v soukromém sektoru, pro kontrolu provozu a především pro prevenci kriminality.

Většina současných systémů potřebuje lidské operátory, aby neustále sledovali dění na monitorech. Jenže počet kamer přerůstá počet operátorů a ti nestíhají všechno sledovat, proto je snaha pomocí umělé inteligence zajistit automatizovaný dohled. Existují systémy pro sledování lidí, automobilů nebo obecně objektů a jejich rozpoznávání, také detekování pohybu a ukládání videjí pro budoucí analyzování. To vše potřebuje velký výpočetní výkon a také velké uložné místo pro ohromný počet dat.

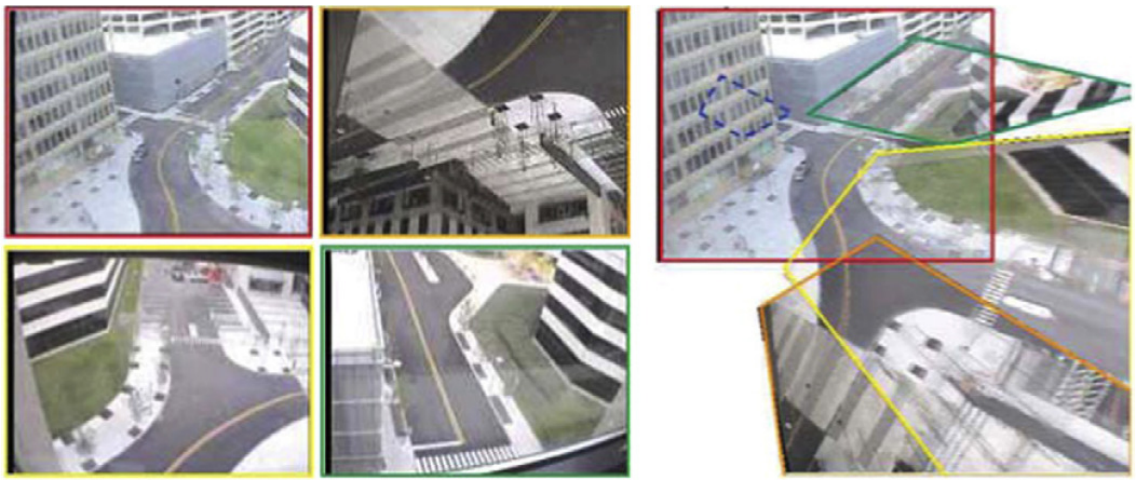
Dohledové kamerové systémy v případě, že patří stejnému majiteli lze sjednotit, tím se sníží náklady a zlepší se škálovatelnost a údržba, protože všechny systémy budou sjednoceny do jednoho rámce a lze je poté spravovat z jednoho místa [14, 6].

Kalibrace kamer

Kalibrace kamer je jedním ze základních problémů v počítačovém vidění a je nezbytná v mnoha dohledových aplikacích. Jde o určení transformace mezi souřadným systémem kamery a scény. Kalibrace spočívá v nalezení parametrů, které vyjadřují zobrazení bodu v prostorové scéně do bodu v rovinném obraze. Parametry jsou vnější a vnitřní. Parametry vnitřní určují transformaci souřadnic bodů do bodů v rovině obrazu. Vnější určují transformaci souřadnic z referenčního souřadného systému do souřadného systému kamery.

Výpočet topologie snímání kamer

Topologie snímání kamer určuje pohledy kamer, jejich překrývání či přiléhání. Mezi pohledy dvou kamer může existovat cesta, která přímo propojuje pohledy těchto dvou kamer. Výsledkem je poté, že objekt, který opouští zorné pole jedné kamery vstupuje do zorného pole jiné kamery. V jistých situacích je to velmi náročné zajistit, aby neexistovala slepá místa.



Obrázek 2.2: Ukázka pohledů kamer (převzato z [14]).

Detekování objektů

Vyhledávání v kamerových systémech se skládá ze dvou částí: intra-kamerového vyhledávání a inter-kamerového vyhledávání. Intra-kamerové vyhledávání je sledování objektů z jedné kamery a inter-kamerové je sledování objektů z vícero kamer. Inter-kamerové vyhledávání je více komplexní než intra-kamerové a je mnohem složitější, především z důvodu možné změny podmínek u každé z kamer např. světla. Nejtypičtější způsob detekce objektů je v 3D souřadném systému nebo v obecné prostorové rovině. Inter-kamerové sledování má velkou výhodu v případě, kdy nastane situace, že detekovaný objekt není ve výhledu jedné z kamer, tak se přepne na jinou kameru s lepším výhledem na daný objekt.

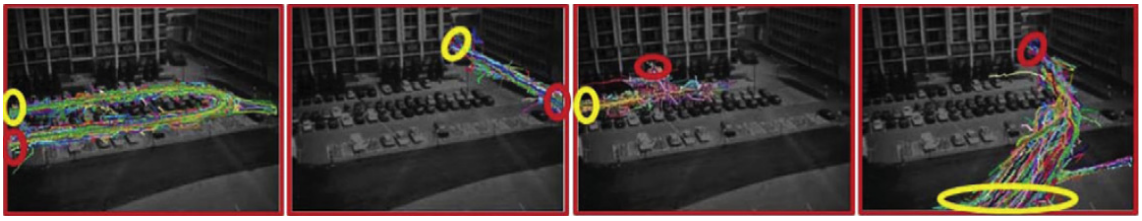
Re-identifikace objektů

V případě že objekt není již v záběru žádné z kamer a po nějakém čase se znovu objeví v záběru, je důležitá re-identifikace objektu, která kompenzuje špatnou topologii kamer.

Vzhled objektu je hlavní vlastnost objektu podle níž se re-identifikace probíhá. Stěžejními parametry vzhledu objektu jsou barva, tvar a textura. Barva má výhodu v tom, že je neměnná bez ohledu na úhel pohledu, ale má nevýhodu v tom, že se může měnit v různých světelných podmínkách nebo jinak vnímaná rozdílným nastavením kamer. Rozpoznávání tvaru může probíhat na základě histogramu orientovaných gradientů, což je diagram se schopností rozpoznat a případně odlišit konkrétní dílčí součásti sledovaného objektu na základě jeho rozložení a proložení jednotlivých částí čtvercovou sítí. Tento proces má dobrou rozlišovací schopnost u drobných pohybů a rotací, ale komplexnější pohyby vyžadují rozsáhlou databázi možných pohybů. Na stanovení textury existují filtry.

Analýza pohybu

Analýza pohybu je základní součástí dohledových systémů. Dohledové kamerové systémy se dají použít také na analýzu pohybu. Tyto systémy se dokáží učit na základě častých trajektoriích a ty rozčlenit do kategorií například pohyb automobilů na parkovišti. Na základě těchto kategorií je systém schopen rozeznat abnormální chování.



Obrázek 2.3: Rozčlenění do kategorií na základě trajektorií (převzato z [14]).

Kapitola 3

Návrh

V této kapitole budou popsána existující řešení inteligetních video-přehrávačů a popis návrhu mé verze přehrávače.

Mým úkolem je nastudovat již hotová řešení inteligentních video-přehrávačů a navrhnout vlastní. Aplikace má umožňovat efektivní použití při vyhledávání užitečných událostí z obrazových záznamů dohledových kamer. Uživatelské rozhraní video-přehrávače má obsahovat jak ovládací prvky přehrávače, tak i zajímavé události z videa, které budou vhodně zobrazovány. Pro realizaci budou použity webové technologie JavaScript s frameworkem JQuery, HTML a CSS.

3.1 Existující řešení

Zde jsou stručně popsány některé z již hotových řešení detekce událostí z videozáznamů dohledových kamer.

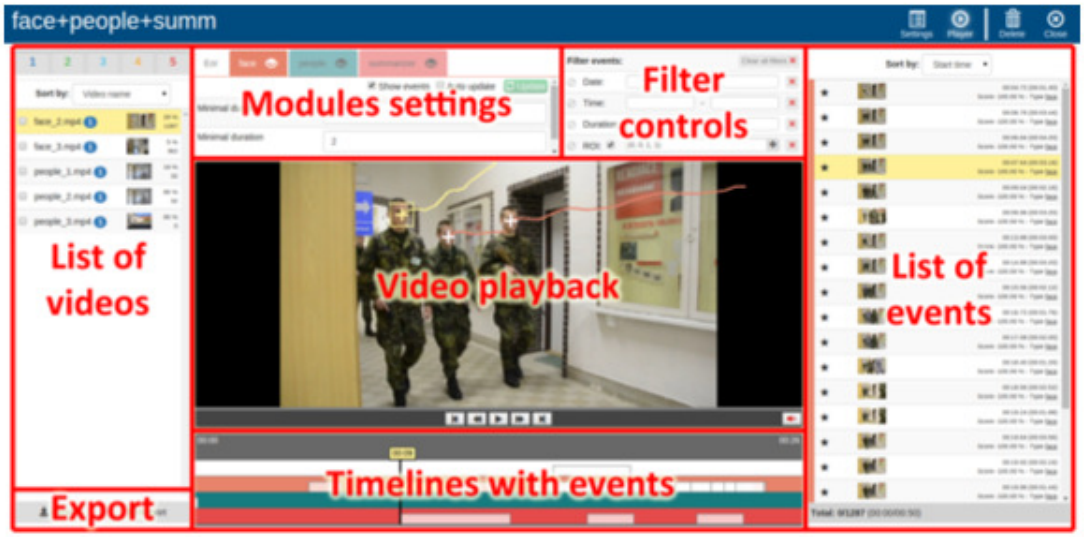
Eyedea

Tato společnost nabízí softwarové nástroje pro detekci a rozpoznávání objektů, které využívají unikátní technologie umělé inteligence a strojového učení. Mezi produkty této firmy patří EyeFace SDK (knihovna pro detekci obličeje), rozpoznávání modelu a výrobce aut, Eyedentity (software pro rozpoznávání obličejů) aj.

Eyedentity umožňuje detekovat obličeje z fotografie či video záznamu, obsahuje deskriptor nezávislý na postoji, osvětlení nebo výrazu ve tváři, podporuje většinu obrazových formátů a využívá podobnostní metriku pro rychlé rozpoznávání z milionů zaznamenaných jednotlivců.

Evidant

Evidant je nástroj pro demonstraci funkcionalit metod vyvinutých v rámci projektu Videoterror. Evidant je inteligentní klient/server videopřehrávač, který zpracovává videa a detekuje v nich zvolené události či prvky. Při vývoji Evidantu probíhaly konzultace s detektivy Policie České republiky a na základě jejich požadavků byly specifikované některé situace vyskytující se ve video datech. Metody vyvinuté v rámci projektu byly upraveny tak, aby splňovaly dané požadavky. Evidant umožňuje uživateli vybrat si obrázky, videa a modul pro jejich zpracování, efektivně procházet zajimavými událostmi, dělat si poznámky na vybrané akce či vybrat akce pro export v již zpracovaných datech [1].



Obrázek 3.1: Ukázka grafického rozhraní přehrávače Evidant (převzato z [1])

3.2 Stanovení cílů

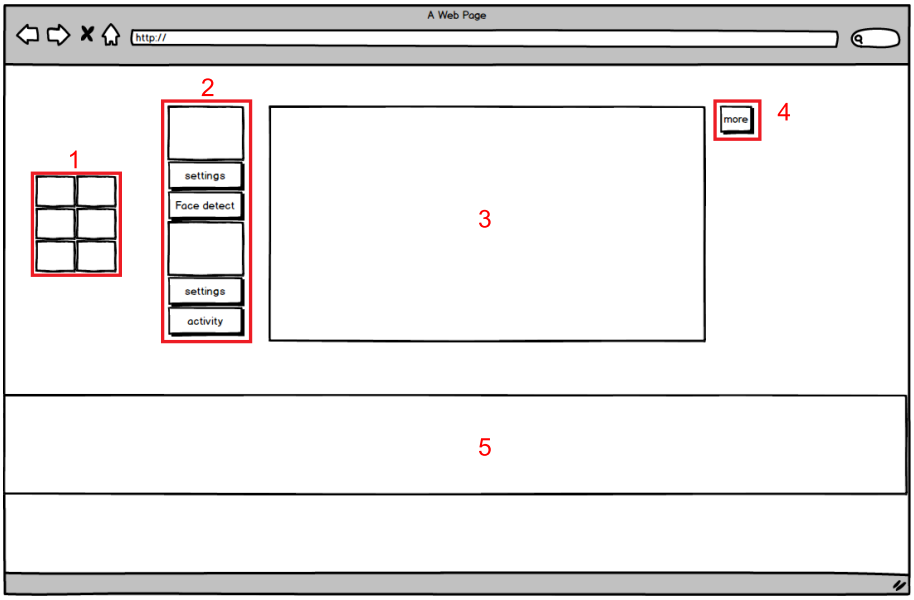
Evidant nevyhledává zajímavé události v reálném čase a to v případě, že člověk potřebuje sledovat vícero kamer a nemůže si dovolit zpoždění, není moc použitelné. Mým cílem je vytvořit video-přehrávač, který bude schopný vyhledávat ve videích uživatelem zvolené události v reálném čase a zobrazovat pouze ty, které jsou zajímavé. Aplikace nebude vytvořena architekturou klient/server, ale bude pouze na klientovi, tudíž bude vyžadován výkon PC sestavy uživatele. Pro demonstraci funkčnosti zobrazování událostí budou použity jednoduché a rychlé algoritmy v jazyce JavaScript pro hledání událostí.

Uživatelské rozhraní této webové aplikace bude responzivní a intuitivní. Bude kladen velký důraz na jednoduchost a funkcionalitu celého rozhraní. Přehrávač bude možné spustit jak na PC, tak i na výkonném tabletu, aniž by uspořádání prvků bylo rozházené či nepoužitelné. Barvy budou zvolené tak, aby uživatel byl schopen sledovat monitor po delší dobu bez nutnosti dát si pauzu z důvodu bolesti očí.

Časová osa v aplikaci bude zobrazována pouze při zvolení modulu, aby zbytečně nerozptylovala uživatelovu pozornost. Časová osa bude vytovořena ve formě náhledů generovaných po spuštění určitého modulu i zde bude kladen důraz na responzivitu. Osa se bude sestávat z náhledů pod kterými budou zobrazována data o čase pořízení daného snímku a také z panelu, která mění barvu na základě detekce událostí. Tato komponenta je stěžejní částí celé aplikace, díky ní by uživatel měl mít možnost sledovat i jiné zařízení nebo dát si pauzu, aniž by přišel o důležité části video-záznamu. Pro možnost odkládání zajímavých snímků bude vytvořena odkladová část, která bude umožňovat snímky zvětšit a také smazat.

3.3 Návrh rozhraní

Grafické rozhraní video-přehrávače bude mít tři varianty závislé na velikosti okna. Verze pro největší okno viz obr. 3.2. Prvky přehrávače jsou očíslované následovně:



Obrázek 3.2: Návrh rozhraní videopřehrávače pro největší okno (min 1794px).

1. odkládací prostor pro zajímavé náhledy - Tento element je důležitou částí aplikace, protože zajímavé snímky mohou zmizet dříve, než budou důkladně prohlédnuté, to také záleží na zvoleném počtu zobrazovaných náhledů. Pro odkladovou plochu bude připraveno 6 boxů pro snímky, které je pak možno zvětšit.

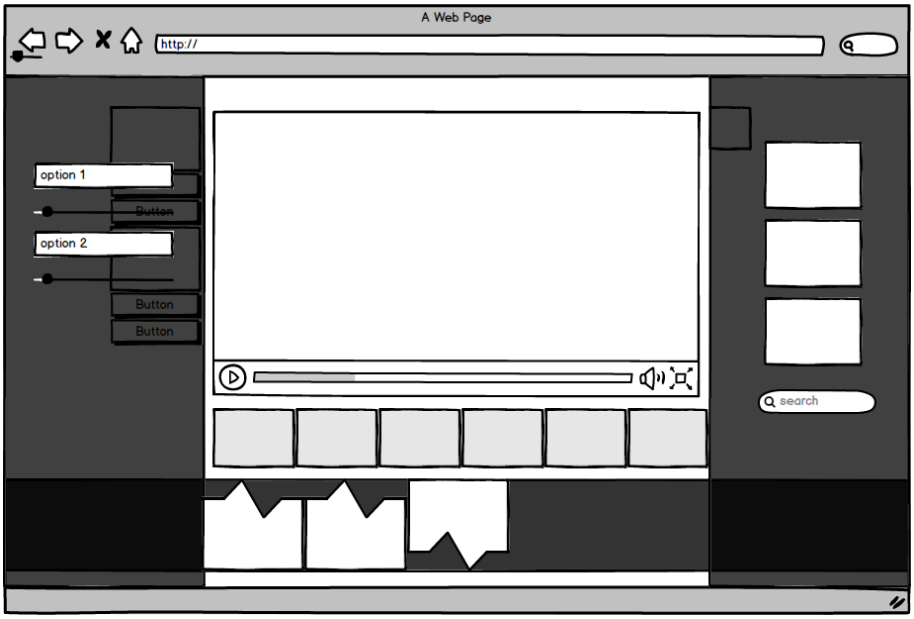
2. oblast s moduly - Zde se nachází moduly pro přehrávač. Aplikace bude mít dva moduly, každý s tlačítkem nastavení a tlačítkem spustit/zastavit. Při stlačení tlačítka nastavení se vysune z levého boku obrazovky panel, na kterém budou zobrazeny možnosti pro nastavení modulu viz obr. 3.3.

3. okno pro zobrazení videa - Na tomto prvku se bude zobrazovat obsah videí a v případě spuštění modulu pro hledání obličeje se zde zobrazí i obdelník ohraničující danou část videa.

4. tlačítko pro zvolení zdroje videa - Toto tlačítko bude fungovat na podobném principu jako tlačítko nastavení v modulech s tím rozdílem, že panel bude vyjíždět z pravé strany okna viz obr. 3.3. V panelu bude nabídka videozáznamů a možnost si vybrat videozáznam vlastní ze svého PC.

5. časová osa - Nejzajímavější prvek přehrávače, na kterém se budou zobrazovat zajímavé náhledy z video záznamů. Na každý náhled bude umožněno kliknout a tím jej přesunout na odkladovou plochu. Za zajímavé náhledy budou považovány ty, které přesahují mezní hodnoty stanovené jako již zajímavé. V případě detekce obličeje se bude jednat o velikost zobrazované plochy a v případě pohybového detektoru o míru změny aktuálního snímku oproti předchozímu.

Rozhraní mého prototypu je inspirováno aplikací Final Cut Pro X, a to především časová osa. Final Cut Pro X je aplikace, která má jiný záměr než můj prototyp, ale pracuje s videem taktéž.



Obrázek 3.3: Návrh vysunovacích panelů pro nastavení aplikace.

3.4 Návrh modulů

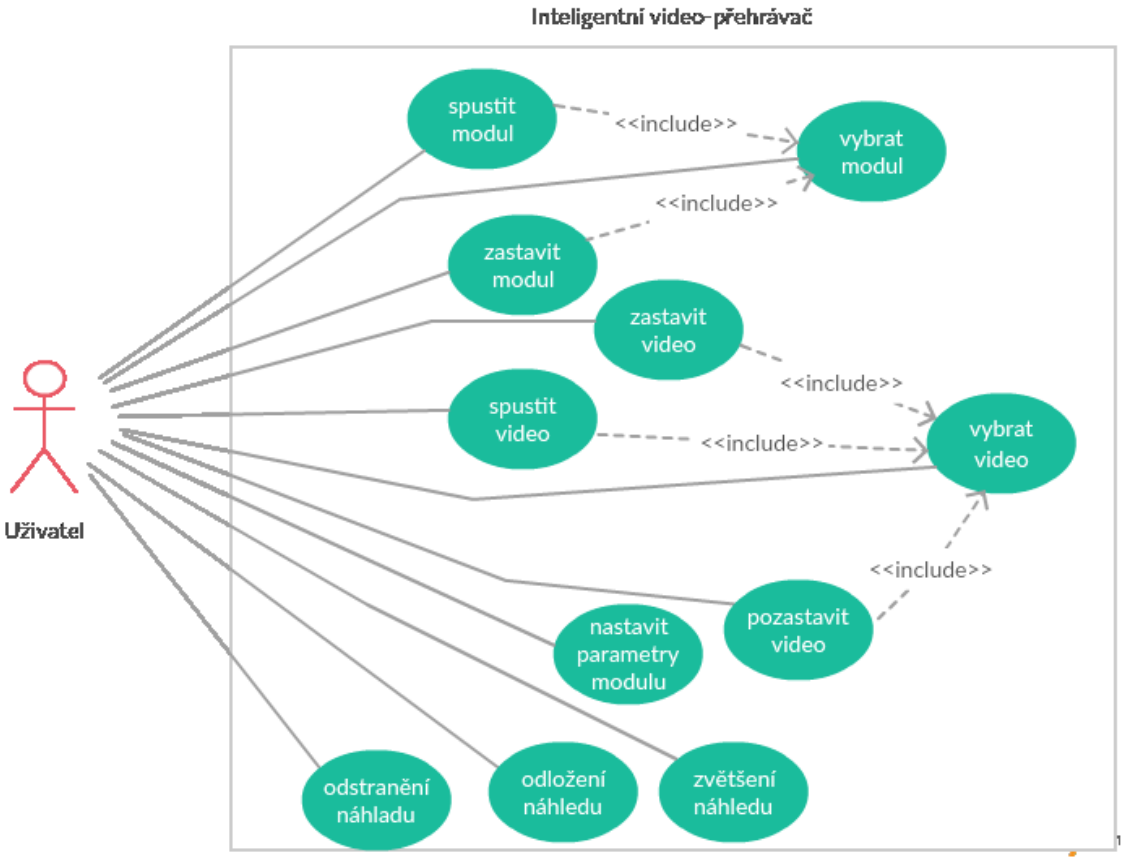
Tento prototyp inteligentního video-přehrávače bude obsahovat dva detektory. Z hlediska největší využitelnosti jsem zvolil detektor obličeje a pohybový detektor. V mém prototypu budou mít oba detektory odlišné použití, každý bude spadat pod jiný typ kamery. V případě detekce obličeje půjde o kameru převážně vnitřní a u pohybového detektoru o kameru venkovní. Vycházel jsem z předpokladu, že kamera pro detekci obličeje bude posazena u bankomatu nebo přepážky a v případě pohybového detektoru bude kamera posazena na méně frekventované ulici, kde se občas stane například nehoda či trestný čin.

Detektory krom odlišného použití budou mít také odlišný způsob prezentace ve video-přehrávači. Obličejový detektor bude vykreslovat na video vymezenou oblast výskytu obličeje a v případě pohybového detektoru budou data zobrazována až na časové ose.

Každý modul nebo-li detektor bude mít své nastavení. U obou modulů bude možné nastavit počet zobrazovaných náhledů na časové ose, než budou postupně zahazovány. Pohybový detektor bude mít navíc volbu prahu, kterým uživatel bude moci měnit mez pro určení zda byl daný pixel změněn nebo nikoli. Bude-li hodnota prahu příliš nízká na časové ose se budou generovat všechny snímky v opačném případě žádné. Při vhodném zvolení prahu je možné zredukovat nežádoucí vlivy např. změnu osvětlení pozadí.

3.5 Případy užití

Webový video-přehrávač bude mít tyto funkce viz obr. 3.4 Na obrázku lze vidět, že každému uživateli bude umožněno si zvolit zdroj video záznamu, vybrat si modul a nastavit ho, používat standardní ovládací prvky pro přehrávání, odložit si náhled pro podrobné zkoumání obsahu a smazat jej.



Obrázek 3.4: Use-case diagram pro video-přehrávač.

Kapitola 4

Implementace

V této kapitole bude pojednáno o použitých technologiích, o postupu při implementaci jednotlivých částí aplikace a celkové realizaci inteligentního video-přehrávače. Tato kapitola je zaměřena na postup, kterým jsem aplikaci vytvářel od základního rozvržení až po jednotlivé elementy přehrávače.

Při vytváření webové aplikace nebo stránky je vhodné postupovat odspodu tzn. (nejdříve struktura, poté design a nakonec funkcionalita), jak již bylo zmíněno v kapitole teorie, i já jsem se této rady držel.

V této kapitole budou popsány použité technologie a také části mého postupu při implementaci inteligentního video-přehrávače: struktura dokumentu, tvoření vzhledu a tvoření funkcionality.

4.1 Technologie

Pro vývoj inteligentního video-přehrávače jsem zvolil knihovnu jQuery, ačkoliv jsem upřednostňoval React.js[1]. Knihovna React.js je mladá a v dnešní době velmi oblíbená, především díky jejímu revolučnímu pojetí implementace jednostránkových webových aplikací. Naproti tomu jQuery je volně dostupná již více jak deset let, má možnost jednoduchého a efektivního použití.

JQuery

JQuery je knihovna, která zjednodušuje interakci s objektovým modelem dokumentu a jazykem JavaScript. Tato knihovna je velmi rychlá a přesná, výrazným způsobem zjednodušuje tvorbu a správu komponent. Použitím jQuery výrazným způsobem redukujeme počet napsaných řádků kódu oproti čistému JavaScriptu. JQuery byla představena v roce 2006 a je volně přístupná. Velkým přínosem této knihovny je výběr jednotlivých elementů v objektovém modelu dokumentu (DOM), a to především díky tomu, že využívá selektorové jádro Sizzle[2]. JQuery také umožňuje rychle a jednoduše měnit elementy modelu DOM, také obsahuje širokou škálu efektů, které je snadné použít. Výčet některých vlastností knihovny jQuery: je volně přístupná, je malá, je velmi oblíbená, je optimalizovaná pro vývoj v moderních webových prohlížečích, je vyvíjena otevřeně tzn. kdokoli může opravovat chyby a vylepšovat či pomáhat s vývojem knihovny, snaží se vyhnout konfliktům s jinými knihovnami pro jazyk JavaScript a mnoho dalších užitečných vlastností. [3, 8]

[1] https://facebook.github.io/react/
[2] https://sizzlejs.com/

4.2 Struktura dokumentu

Tuto část jsem implementoval v jazyce HTML. Na začátku je důležité nezapomnout na zásady, a to na kódování, použitou verzi jazyka HTML, správnou strukturu dokumentu a také vše mít označkované. Po napsání základní struktury dokumentu jsem rozdělil tělo dokumentu do větších částí (video, moduly a časovou osu), ale také na menší (odkladovou plochu a vysunovací volby). Tyto části tvoří tělo dokumentu. Nyní větší části popíši o něco blíže.

Začneme částí video, tato část obsahuje video element tvořený značkou `<video>`, kterou je možné používat od verze 5 jazyka HTML. Video element lze použít již s předdefinovanými ovládacími prvky, této možnosti jsem nevyužil a vytvořil si vlastní ovládací prvky. Ovládací prvky jsem vytvořil pomocí prvků input typu submit. Část video obsahuje také dvojici pláten tedy canvas elementy tvořené značkou `<canvas>`, které jsou nutné pro funkcionalitu inteligentního video-přehrávače. Celá video část si bude moci měnit pozici s modulovou částí.

Část moduly je složena ze dvou modulů, které se sestávají ze stejných elementů, a to z elementu figure, ve kterém je obrázek daného modulu a tlačítek settings a start/stop. Tato tlačítka jsou vytvořena pomocí elementu input typ button.

Časová osa na počátku neobsahuje nic zvláštního, je tvořena bloky do nichž se budou po spuštění videa a modulu vkládat náhledy a data. Tato část bude více popsaná v podkapitole Tvoření funkcionality.

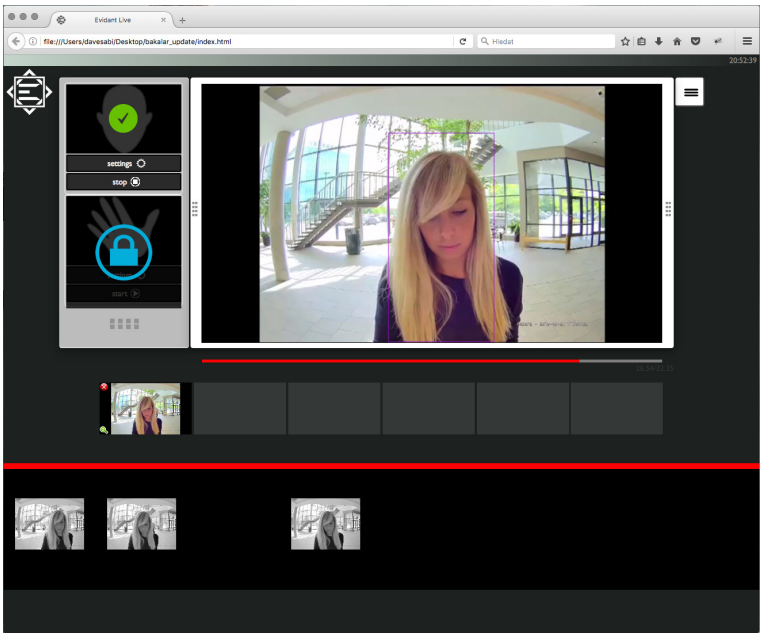
4.3 Tvoření vzhledu

Vzhled je důležitá část celé aplikace, při vytváření vzhledu jsem hledal správné kombinace barev a inspiroval jsem se tmavým prostředím aplikace Spotify[3]. K vytváření kvalitního vzhledu, který nejedná na úkor funkčnosti ve webových aplikacích či stránkách jsem použil kaskádové styly. Z počátku jsem používal framework Bootstrap[4] pro responzivní design aplikace, ovšem po nějaké době jsem se rozhodl tohoto frameworku zanechat z důvodu velkého množství vaty a díky tomu zbytečně nekomfortnímu aplikování pro mé potřeby.

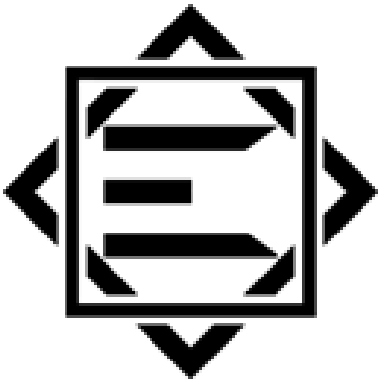
Na obrázku 4.1 je zobrazena aplikace v aktivním stavu. Grafické uživatelské rozhraní má tři verze zobrazení v závisloti na velikosti okna. Této skutečnosti jsem dosáhl použitím pravidla `@media` typu screen, kde jsem nastavil požadované rozmezí velikostí pro dané verze. Do každého pravidla media stačilo napsat požadované deklarace stylů, které budou zobrazeny v tomto rozmezí pixelů. Responzivitu samotných prvků, například náhledů v časové ose, jsem implementoval pomocí zobrazení typu flex. V případě, že kontejner je nastaven na zobrazení flex, tak prvkům uvnitř je možné nastavit velikost vůči ostatním pomocí hodnoty flex. V případě, že všechny prvky uvnitř kontejneru mají stejnou hodnotu flex, pak jsou zobrazeny ve stejném poměru. K docílení efektu zvětšení po najetí na náhled je hodnota flex zvětšena oproti ostatním, a tím se poměr změní.

Jednoduchost a intuitivnost grafikého rozhraní aplikace jsem realizoval tak, že nastavení přehrávače a prvků, které nejsou potřeba vidět po celou dobu práce s aplikací, jsou zobrazovány pomocí vysunovacich panelů, které neruší uživatele a aplikace pak působí čistě a uživatelsky přívětivě. Ovládací panel pro přehrávač je taktéž schován a je zobrazen v případě najetí kurzoru na přehrávač.

[3] https://www.spotify.com/cz/
[4] http://getbootstrap.com/



Obrázek 4.1: Ukázka apliakce v aktivním stavu.



Obrázek 4.2: Logo prototypu inteligentního video-přehrávače.

4.4 Tvoření funkcionality

Tato podkapitola popisuje způsob implementace důležitých funkcionalit přehrávače.

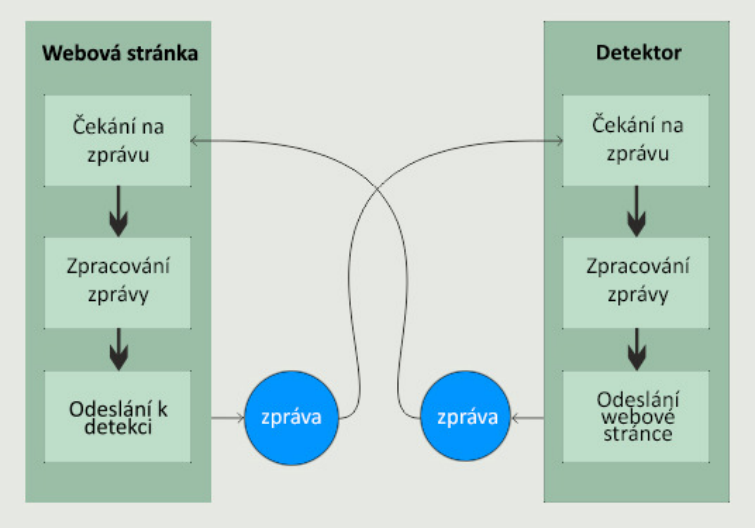
Ovládací prvky

Ovládací prvky pro přehrávání a pro moduly. Tyto prvky fungují na principu naslouchání událostí, konkrétně události kliknutí. Ovládací prvky přehrávače jsou přehrát, pozastavit, zastavit, vrátit a vypnout či zapnout zvuk. Co se týče ovládání modulů, tak moduly mají ovládání v podobě nastavení a spustit či zastavit modul. Pro předcházení chybových stavů je vykreslen zámek v situacích, které by uživatele mohly zmást nebo přehrávač ochromit.

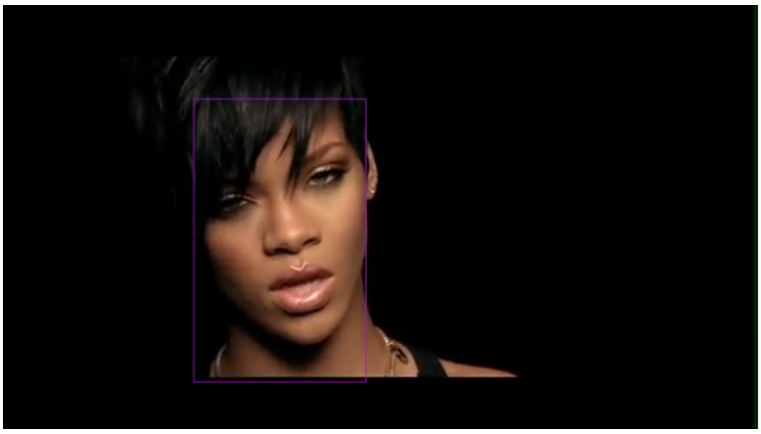
Vyznačení oblasti na videu

Princip vyznačování oblasti na videu spočívá v implementování dvou pláten. Na jedno plátno je vyreslován obsah videa, toto plátno není zobrazováno a je z něj pouze vytahován obsah, který je posílán v podobě zpráv modulům. JavaScript umožňuje z plátna vzít obsah a také jej analyzovat či pozměnit. V případě mého prototypu se jedná o situaci, kdy po časových intervalech je z videa překreslen obsah na plátno, tento obsah je poté přeposlán spolu s výškou a šířkou plátna modulu pro detekci obličeje. Tento modul poté pošle nazpět souřadnice vrcholů obdelníku. Tyto souřadnice jsou poté použity na zakreslení obdelníku na druhé plátno, které je srovnáno přesně podle videa, aby fungovalo jako pomyslná fólie.

Snímky se přeposílají pomocí zpráv z nutnosti implementace více-vláknového přístupu, protože práce detektoru zabírá velké množství výkonu PC a na používání video-přehrávače pak nezbývá skoro žádný výkon. Verze přehrávače bez více-vláknového přístupu byla nepoužitelná, protože na zbylé operace přehrávače nezbyl již potřebný výkon a aplikace se zasekávala. Více-vláknový přístup umožnilo aplikování tzv. webových dělníků (Web Workers). Web Workers jsou API, které webovým vývojářům dovolují oddělit náročné skripty od stránky. Tyto skripty pracují na pozadí a nemají přístup k obsahu stránky, komunikace s nimi je umožněna pomocí zpráv.



Obrázek 4.3: Model komunikace webové stránky s detektory prostřednictvím Web Workeru.



Obrázek 4.4: Ukázka vymezení detekované části.

Moduly

Veškeré snahy jak využít výkon PC k zajištění dostatečně rychlého detekování obličeje ztroskotaly na detektorech. V aplikaci jsem vyzkoušel řadu existujících detektorů (tracking.js, js-objectdetectkt) a ani jeden rychlostně nestačil, z tohoto důvodu jsem byl nucen tyto detektory implementovat sám ve zjednodušené podobě. Nakonec jsem použil algoritmus z knihy [11]. Detektor obličeje je jednoduchý algoritmus, který porovnává barvu pixelů s nadefinovanou barvou pleti. Oblast s nejvyšším počtem těchto pixelů je zvolena jako výsledná a nastaví se souřadnice vrcholů pro vykreslení obdelníku. Tento algoritmus je rychlý, ale bohužel nepřesný. Velké problémy zde dělá pozadí, které může mít podobnou barvu jako pleť člověka a také to, že algoritmus detekuje pouze jeden objekt. V algoritmu chybí detekce očí a úst, které by výsledek zpřesnily. Tento modul je určen především pro kamery u bankomatů nebo přepážek. V případě vylepšení nebo výměny tohoto detektoru za lepší by nebyl problém použít tento detektor na klasické venkovní kamery. Pseudo kód pro detekci obličeje:

```
procházej pixely ve snímku
  proveď kontrolu zda jde o barvu pokožky
    pokud ano nastav souřadnice regionu a nastav velikost regionu na 1
procházej pole regionů
  pokud je velikost regionu větší než 0
    přidej pixelům 5x5 distance a zapamatuj si velikost
    obdelníkové oblasti, kterou pokrývají
procházej regiony
  pokud má region větší velikost než 0
    nastav jej jako dosud největší a opakuj
```

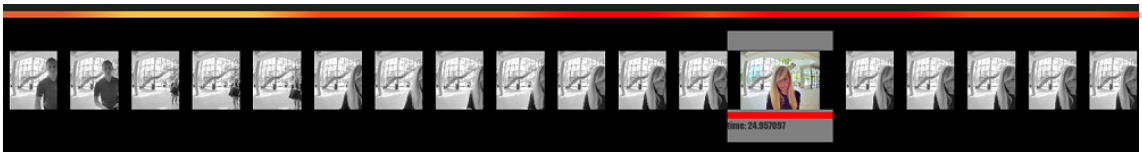
Pomocí knihy [11] jsem naimplementoval i detektor pohybu. Pro detekci jsem použil rozdílovou metodu. Tato metoda spočívá v porovnávání snímků z jiných časových okamžiků. Tuto metodu jsem naimplmentoval tak, že snímek je nejdříve převeden na stupně šedi a poté jsou porovnávány předchozí a aktuální snímek. Pokud je rozdíl větší než-li zadaný práh, pak je tento pixel považován za změněný. Počet těchto změněných pixelů je zaznamenáván

a následně převeden na procenta. Nevýhodou této metody je měnící se pozadí ve snímcích, to ovlivňuje výrazným způsobem výsledek detekce.

Oba detektory nemohou pracovat současně, protože výpočetní náročnost je velká a dochází poté k zpomalení a v nejhorším případě spadnutí celé aplikace. Výpočetní náročnost způsobuje především rozebírání pixelů z plátna, které klade velký nárok na webový prohlížeč. Většina webových prohlížečů nepodporuje či zakazuje brát pixely z obrazu, ovšem prohlížeč Mozzila Firefox toto umožňuje. Celá aplikace je tedy směřována k tomuto prohlížeči a je funkční právě na něm.

Časová osa

Časová osa je ztěžejní prvek apliakce, z počátku byla snaha jejího vytvoření pomocí grafu, ale tento způsob řešení se později ukázal jako nevhodný. Zvolil jsem tedy variantu náhledů, tato varianta má výhodu v tom, že uživatel vidí ihned v jakém snímku se stala zajímavá událost, dopomáhá tomu i úzký panel, který mění barvu pomocí gradientu v závislosti na míře pohybu, v případě pohybového detektoru a v případě detektoru obličeje velikost plochy, kterou obličej zabírá. Osa umožňuje na náhled snímku kliknout, ten se poté přesune na odkladovou plochu, kde je možné s ním dále pracovat.



Obrázek 4.5: Ukázka časové osy pro detekci obličeje.

Princip jakým jsem tohoto docílil. Náhledy jsou generované po časových intervalech stejných jako detektor. Princip generování je obdobný jako v případě vykreslení obsahu videa na plátno pro detekci. Náhledy jsou taktéž plátna, to má i nevýhody, protože plátno nelze duplikovat. Pro přesunutí náhledu na odkladovou plochu se nejedná o problém, protože se náhled přesune, ale v případě zvětšení, které umožňuje odkladová plocha již problém nastává, ale o tom až v části Odkladová plocha. Při každém intervalu se generuje nejen náhled, ale také panel do kterého se náhled vloží, taktéž paragraf s daty, neviditelný snímek originální velikosti a barevný pruh znázorňující míru změny či velikosti obličeje v daném snímku. Tento celek je pak vykreslen v časové ose, data jsou zobrazována až při najetí kurzoru na náhled. Časové osy jsou dvě, každá pro jiný modul. Z hlediska šetření uživatelovi pozornosti a místa na obrazovce, je zobrazena pouze jedna a to ta, která patří aktivnímu modulu. Po změně modulu se časové osy prohodí, ale uchovají si své vytvořené náhledy.

Obrázek 4.6: Ukázka časové osy pro pohybový detektor.

Odkladová plocha

Tato část aplikace vznikla, jako následek zjištění ze zkoušení aplikace, že při nastavení malého počtu náhledů a v případě, že uživatel důkladněji zkoumá snímek, může pak přijít o další zajímavé události, které mezitím zmizí. Odkladová plocha tedy slouží k odložení náhledů pro dodatečné zkoumání.

Odložený náhled je možné smazat nebo zvětšit. Jak bylo zmíněno v předešlé části, plátna se nedají duplikovat a zvětšení zde funguje na způsobu překreslení onoho neviditelného originálního snímku na nové plátno. Tato operace se provede pokaždé, kdy uživatel zaklikne zvětšení náhledu. Snímek v originální velikosti je smazán po uzavření modulového okna.

Obrázek 4.7: Ukázka odkladové plochy pro náhledy.

Kapitola 5

Testování

Tato kapitola je zaměřená na testování uživatelského rozhraní aplikace. Jsou zde popsány použité metody a můj postup při testování.

5.1 Použité metody

Pro testování existuje řada metod, já jsem použil experimentální metodu a metodu pozorování. Uživatelům by mělo být umožněno projevit své názory bez ovlivňování pozorovatelem, pro tento účel jsem použil dotazník [12].

Metoda pozorování

Metoda pozorování má dva typy, přímé pozorování a nepřímé. V případě přímého pozorování si pozorovatel píše poznámky, případně měří čas. Velkou výhodou je, že pozorovatel dostává zpětnou vazbu od testujícího bezprostředně poté, co ukončil provádění úkolů. Pozorovatel je také schopen určit, která část úkolu dělala testujícímu největší potíže. Tato metoda má ovšem i stinné stránky. Testující se nemusí cítit uvolněně při sledování pozorovatelem v průběhu plnění úkolů, tato skutečnost může ovlivnit výsledek testování. Je také důležité, aby si pozorovatel jasně stanovil na co se zaměří při sledování testujícího, aby tak zabránil případnému přehlédnutí důležitých informací.

Nepřímé pozorování spočívá v tom, že testující není pozorován přímo pozorovatelem, ale jeho úkony jsou zaznamenávány například video kamerou. Tato metoda může působit na uživatele také nepříjemně. Pocit nepříjmeného sledování je tu podobný jako v předchozím případě. Nepřímé pozorování je možné aplikovat na krátké sekvence nebo na dlouhé nahrávky po dobu několika týdnů. Anylyzování bývá časově náročné. Je možné problémy analyzovat, lokalizovat a případně i opravit.

Experimentální metoda

Pomocí experimentů je možné sledovat vlivy proměnných na stávajícím řešení. Mezi proměnné patří např. doba provedení úkolu, věk či pohlaví uživatele aj. Určení vlastností důležitých pro daný experiment záleží na návrhu experimentu.

Průběh testování

Testování jsem zahájil použitím metody pozorování, a to varianty přímého pozorování. Testující byli jak lidé zdatní v informačních technologiích, tak i lidé jiného zaměření. S tes-

tujícími jsem se osobně setkal a průběh testování jsem jim důkladně vysvětlil. Testující se měli cítit uvolněně a zapomenout na to, že jsou pozorováni. Tento způsob testováni jsem aplikoval na osmi lidech různého stáří a zaměření. Všichni testující měli stejné úkoly a podmínky. Seznam úkolů pro testující:

1. vybrat si zdroj video-záznamu a spustit jej
2. vybrat si modul, spustit a nastavit ho
3. přepnout na druhý modul a nastavit ho
4. odložit náhled na odkladovou plochu
5. zvětšit náhled a následně jej smazat
6. vybrat si jiný video-záznam ze složky v počítači

Po aplikaci metody pozorování jsem rovnou přešel k experimentu. Cílem tohoto experimentu byla míra zvýšení rychlosti a použitelnosti uživatelského rozhraní po druhém použití aplikace stejným testujícím. Zde jsem sledoval převážně dobu potřebnou k dokončení daného úkolu a vliv technického zaměření testujícího na rychlost používání aplikace.

5.2 Analýza a vyhodnocení

Pozorováním jsem vysledoval, že většina testujících tápe při prvním úkolu. Předpokládám, že je na vině málo viditelné tlačítko nebo málo zažitý styl tlačítka pro nastavení zdroje. Menší tápání jsem zaregistroval při výběrech modulu, kdy testující nevěděli, že video se nejdříve musí spustit a až poté se dá nastavit a aktivovat modul. Práce s časovou osou byla naopak velmi zdařilá a všichni testující dokázali úkoly týkající se právě časové osy zvládnout rychle a hned na první pokus.

Z této metody usuzuji, že uživatelské rozhraní není chaotické a uživatelé jsou po krátké době schopni s tímto rozhraním pracovat efektivně. Experimentální metodou, která na-

pokus č.	pohlaví	IT vzdělání	úkol č. 1 /s	úkol č. 2 /s	úkol č. 3 /s	úkol č 4 /s
1	žena	ne	31	17	6	12
2	-	-	12	3	6	2
1	muž	ano	13	30	7	5
2	-	-	3	5	5	3
1	žena	ne	44	50	13	3
2	-	-	4	7	10	3
1	muž	ano	11	13	6	7
2	-	-	5	5	6	3
1	muž	ne	19	9	7	5
2	-	-	5	6	7	4

Tabulka 5.1: Tabulka znázorňující délku času pro vykonání úkolů 1-4

vazovala na výsledky z pozorování jsem zjistil, že muži s IT vzděláním byli schopni toto uživatelské rozhraní používat rychle již od počátku. Pozitivním zjištěním je, že všichni uživatelé po druhém provádění úkolů byli stejně nebo podobně pohotoví, jako muži s IT vzděláním. Experimentem jsem zjistil, že je pro každého snadné si zvyknout na rozložení prvků i ovládání.

pokus č.	pohlaví	IT vzdělání	úkol č. 5 /s	úkol č. 6 /s
1	žena	ne	10	7
2	-	-	3	4
1	muž	ano	7	7
2	-	-	3	5
1	žena	ne	10	9
2	-	-	4	6
1	muž	ano	15	8
2	-	-	5	7
1	muž	ne	11	13
2	-	-	5	7

Tabulka 5.2: Tabulka znázorňující délku času pro vykonání úkolů 5-6

5.3 Dotazník

Respondentům jsem pokládal otázky, jejichž odpovědi mi pomohou k budoucímu vylepšení aplikace. Dotazník obsahoval pouze pět otázek. Na první tři otázky se odpovídalo pomocí stupnice od 1-10, kde stupeň 10 bylo nejvyšší a nejlepší možné hodnocení. Otázky 4 a 5 byly typu dotazu a odpovědi. Tyto otázky byly pro mě nejdůležitější. Dotazoval jsem se deseti respondentů. Odpovědi byly velice zajímavé a přínosné.

Otázka č. 1

Jak Vám vyhovuje ovládání přehrávače? Na tuto otázku odpověděli uživatelé vesměs pozitivně. Plný počet bodů nebyl dosažen zřejmě z důvodu nevhodně zvoleného tlačítka pro volbu zdroje videa.

Otázka č. 2

Jak hodnotíte celkový design aplikace? Design dopadl nejlépe, všichni respondenti hodnotili design vysokými čísly. Úspěch připisuji zvolené barevné kombinaci a výsuvným panelům.

Otázka č. 3

Jak hodnotíte výkonnost detektorů? Detektory nedopadly vesměs moc dobře. Tento částečný neúspěch připisuji nedokonalým detektorům, a to především kvůli potřebě co největší rychlosti detekce.

Otázka č. 4

Co byste v aplikaci změnili? Na tuto otázku byl velký počet zajímavých nápadů. K těm nejzajímavějším patří:

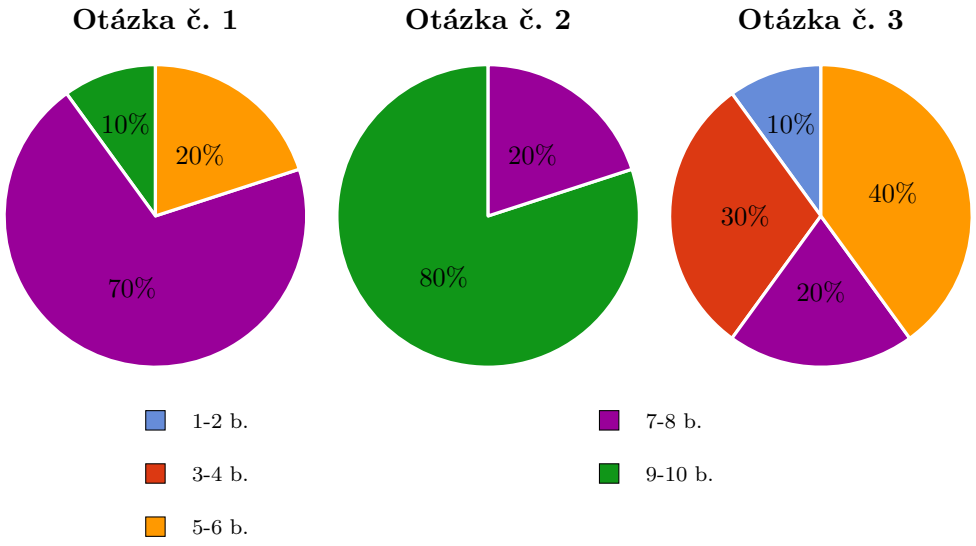
- Při změně modulu automatické zastavení aktuálního modulu.
- Menší obrázky popisující moduly.
- Tlačítko pro volbu zdroje zobrazit až v případě malého okna, jinak zdroje zobrazit přímo na stránce.

- Při zvětšení okna zvětšit i video.

Otázka č. 5

Co byste do aplikace přidali? I zde padly zajímavé nápady. K těm nejzajímavějším patří:

- Možnost zvolení velikosti odkladové plochy.
- Při najetí kurzoru na posuvník na časové ose videa zobrazit malé náhledy.
- Přidání možnosti změny barevné palety webové stránky.
- Přidání databáze do aplikace pro opakované shlédnutí.



Kapitola 6

Závěr

Cílem mé práce bylo prostudovat řešení inteligentních video-přehrávačů a navrhnout vlastní řešení. Při studiu jsem se nejvíce zaměřil na video-přehrávač Evidant a snažil jsem se o vytvoření přívětivějšího grafického rozhraní pro uživatele. Inteligenci video-přehrávače jsem pojal z hlediska funkčních modulů a z uživatelského rozhraní.

Detektory nejsou bezchybné, zde by bylo určitě lepší použití již hotových řešení. Při implementování jsem vyzkoušel knihovny tracking.js a js-objectdetectkt. Obě tyto knihovny nesplnily mé očekávání, jejich rychlost nebyla dostatečná a video-přehrávač pak nebyl použitelný. Implementace pomocí jQuery a čistého JavaScriptu nebyla špatná volba, ačkoli použití React.js by bylo nejspíše výhodnější z hlediska budoucí údržby či vylepšení.

Časová osa vytvořená pomocí náhledů se ukázala jako dobrá volba, v testování to byla nejúspěšnejší část aplikace. Na jednoduchý a čistý design byly taktéž pozitivní ohlasy. Díky použití více-vláknového přistupu pomocí Web Workerů nebude velký problém se změnou modulů či implementování modulů dalších.

Do prototypu je možné přidat vícero detektorů. Pro reálné užití tohoto prototypu by bylo žádoucí naimplmentovat možnost oddělení obrazu videa na jeden monitor a časovou osu na druhý monitor. Díky této operaci by bylo možné časovou osu upravit na větší náhledy a zobrazovat i více informací. Pro aktivování většího počtu detektorů současně by bylo vhodné tyto detektory již přesunout na server z důvodu velkého nároku na výkon PC.

Literatura

[1] Beran, V.; Kapinus, M.; Klicnar, L.; aj.: *VideoTerror demonstrator*. FIT VUT v Brně, Srpen 2015, [Online; navštíveno 20.04.2017].
URL http://www.fit.vutbr.cz/units/UITS/pubs/tr.php.cs?file=%2Fpub%2F11035%2F02_vtdemo-evidant.pdf&id=11035

[2] Budd, A.; Moll, C.; Collison, S.: *CSS filtry: hacky a pokročilé postupy*. Brno: Zoner Press, 2007, ISBN 978-80-86815-54-1.

[3] jQuery Community Experts: *jQuery: kuchařka programátora*. Brno: Computer Press, 2010, ISBN 978-80-251-3152-7.

[4] Dostál, M.: *Základy tvorby uživatelského rozhraní*. Katedra informatiky univerzita palackého v Olomouci, Srpen 2015, [Online; navštíveno 19.04.2017].
URL https://phoenix.inf.upol.cz/esf/ucebni/gui-dostal.pdf

[5] Kosek, J.: *HTML tvorba dokonalých WWW stránek*. Praha: Grada Publishing, 1998, ISBN 80-7169-608-0.

[6] Liu, J.; Nishimura, S.; Araki, T.: Wally: A Scalable Distributed Automated Video Surveillance System with Rich Search Functionalities. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, New York, NY, USA: ACM, 2014, ISBN 978-1-4503-3063-3, s. 729–730, doi:10.1145/2647868.2654872.
URL http://doi.acm.org/10.1145/2647868.2654872

[7] Marcotte, E.: *Responsive web design*. New York: A Book Apart, 2011, ISBN 978-0-9844425-7-7.

[8] Margorín, M.: *jQuery bez předchozích znalostí*. Brno: Computer Press, 2011, ISBN 978-80-251-3379-8.

[9] Meyer, E. A.: *CSS-kompletní průvodce*. Brno: Zoner Press, 2007, ISBN 978-80-86815-64-0.

[10] Meyer, E. A.: *Smashing CSS : professional techniques for modern layout*. Chichester: Wiley, 2011, ISBN 978-0-470-68416-0.

[11] Pfeiffer, S.: *HTML - audio a video*. Brno: Zoner Press, 2011, ISBN 978-80-7413-147-9.

[12] Preece, J.: *Human computer Interaction*. Massachusetts: Addison-Wesley, 1996, ISBN 0-201-62769-8.

[13] Rybička, J.: *HTML5 a CSS pro webové designéry*. Brno: Zoner Press, 2011, ISBN 978-80-7413-166-0.

[14] Wang, X.: Intelligent multi-camera video surveillance: A review. *Pattern Recognition Letters*, ročník 34, č. 1, 2013: s. 3 – 19, ISSN 0167-8655, doi:https://doi.org/10.1016/j.patrec.2012.07.005, extracting Semantics from Multi-Spectrum Video.
URL http://www.sciencedirect.com/science/article/pii/S016786551200219X

[15] Wempen, F.: *HTML a CSS krok za krokem*. Brno: Computer Press, 2007, ISBN 978-80-251-1505-3.

[16] Wigdor, D.; Wixon, D.: *Brave NUI World*. Burlington: Morgan Kaufmann, 2011, ISBN 978-0-12-382231-4.

[17] Wyke-Smith, C.: *Využijte kaskádové styly naplno!* Brno: Computer Press, 2006, ISBN 80-251-1297-7.

[18] Yank, K.; Adams, C.: *Začínáme s JavaScriptem*. Brno: Zoner Press, 2007, ISBN 978-80-86815-94-7.

Přílohy

Příloha A

Obsah CD

K bakalářské práci je přiloženo CD s touto adresářovou strukturou:

- videoPlayer/ - Složka obsahující zdrojové kódy webové aplikace.
- soubor SabelaDavidBp.pdf - Tento dokument.
- soubor prezentačníVideo.mp4 - Video prezentující tuto práci.
- soubor README.txt - Stručný návod k použití aplikace.