Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Janša

# Z anotacijami obogateno predvajanje videa v tehnologiji HTML 5

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Danijel Skočaj

Ljubljana 2012

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

*Tržaška 25*
*1001 Ljubljana, Slovenija*
*telefon: 01 476 84 11*
*01 476 83 87*
*faks: 01 426 46 47*
*01 476 87 11*
*www.fri.uni-lj.si*
*e-mail: dekanat@fri.uni-lj.si*

Št. naloge: 00336/2012
Datum: 04.09.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **NEJC JANŠA**

Naslov: **Z ANOTACIJAMI OBOGATENO PREDVAJANJE VIDEA V TEHNOLOGIJI HTML 5**

**VIDEO PLAYBACK AUGMENTED WITH ANNOTATIONS USING HTML 5 TECHNOLOGY**

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Z razmahom digitalnih tehnologij se je produkcija in uporaba videa zelo razmahnila. Zelo se je razširilo področje uporabe, še bolj pa množica uporabnikov. Hkrati z vse večjo priljubljenostjo in uporabnostjo videa se pojavljajo tudi nove zahteve za video predvajalnike. V diplomski nalogi razvijte aplikacijo, ki bo obogatila predvajanje videa s selektivnim prikazovanjem anotacij v obliki posameznih likov ali besedila. Te anotacije, ki nosijo dodatno informacijo o videu in bodo podane v posebej oblikovani datoteki, naj se izrisujejo preko videa, uporabniški vmesnik pa naj omogoča udobno upravljanje tako z videom kot tudi s samimi anotacijami. Zato, da bo aplikacija dosegljiva čim večjemu obsegu uporabnikov, jo implementirajte v tehnologiji HTML 5.

Mentor:

doc. dr. Danijel Skočaj

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

# Izjava o avtorstvu diplomskega dela

Spodaj podpisani Nejc Janša, z vpisno številko **63060091**, sem avtor diplomskega dela z naslovom:

*Z anotacijami obogaten prikaz videa v tehnologiji HTML 5*

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Danijela Skočaja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 2. oktobra 2012 Podpis avtorja:

*Zahvaljujem se mentorju, dr. Danijelu Skočaju ter še posebej asistentu mag. Luki Čehovinu za vso pomoč, nasvete, popravke ter potrpežljivost. Rad pa bi se zahvalil tudi moji ženi Ani za podporo, tudi staršema in bratom ter sestrama.*

Diplomo posvečam najinemu, še nerojenemu otroku.

# Kazalo

# Seznam uporabljenih kratic in simbolov

- HTML 5 – Hyper Text Markup Language 5
- CSS 3 – Cascading Style Sheets 3
- AJAX – Asynchronus JavaScript and XML
- JSON - JavaScript Object Notation
- CS5 – Creative Suite 5
- HEX - Hexagonal
- RGB – Red Green Blue
- Ogg – Free Open Container Format
- WebM – Open Web Media Project
- H.264 – A standard for Video Compresion
- HTTP – HyperText Transfer Protocol
- XML – Extensible Markup Language
- JSONP - JavaScript Object Notation with Padding
- FPS – Frames Per Second

# Povzetek

Diplomska naloga razloži in pokaže interaktivno obogaten prikaz video anotacij. V diplomski nalogi so opisane tehnologije, orodja, funkcionalnosti in storitve, ki smo jih uporabili pri razvoju. Video predvajalnik in prikaz delovanja je osnovan v spletnih tehnologijah `HTML5`, `JavaScript` , `JQuery`, `CSS` in vmesnika `ajax`. Video predvajalnik ima osnovne funkcije, vsebuje pa prikaz anotacij, dodajanje besedila na plasti videa ter sledenje anotacijam preko miškinega kurzorja. Poudarek diplomske naloge je prikazovanje anotacij preko sledilnih algoritmov, hkrati pa je to uporabno orodje, ki je uporabniku prijazno.

Ključne besede: Spletne tehnologije, video anotacije, HTML5, aplikacijski programski vmesnik

# Abstract

This thesis describes interactively enriched display of video annotations. It contains descriptions of technologies, tools, functionality and services that were used in the developement. Video Player and display of moving pictures is based in web technologies, such as `HTML5`, `JavaScript` , `JQuery`, `CSS` and `ajax` interface. Video Player has basic functions and provides display of annotations, adding text to the video layer and tracking of annotations by the mouse cursor. The focus of the thesis is to show annotations through tracking algorithm, at the same time it is a useful tool that is user friendly.

Keywords: Web technologies, video annotations, HTML5, Application Programming Interface

# Poglavje 1

# Uvod

## 1.1 Motivacija

Zadnja leta se je svetovni splet s povečano uporabo računalnikov in izboljšanjem prenosa podatkov razširil. Z uporabo novih spletnih tehnologij so se možnosti uporabnikov, da lahko izbirajo nove aplikacije, še povečale. Tako v svetu spletnih tehnologij opažamo vse večji porast multimedijskih aplikacij. S predstavitvijo novih spletnih tehnologij so se odprle nove obsežnosti, med njimi tudi risanje dvo- in tridimenzionalnih grafik v grafičnem elementu *canvas*. Preko grafičnega elementa *canvas* pa lahko rišemo tudi video anotacije.

Video anotacije so vizualno bogate oblike, ki se pojavljajo v video posnetkih ob določenem času, ko se posnetek predvaja. Med video anotacije sodijo podnapisi, liki v vektorskih oblikah, slike, internetne povezave, reklame ter tudi anotacije v povezavi s semantičnim spletom. Video anotacije pomagajo uporabnikom, da lažje opazijo določene stvari, vendar jih je potrebno uporabljati na pravilen način. Če so uporabljene napačno, lahko uporabnika zavedejo ali pa mu onemogočijo, da ugotovi njihov namen.

Zahteva diplomske naloge je z uporabo tehnologij *HTML 5* in *Javascript* izdelati uporabniški vmesnik, ki omogoča interaktivni prehod skozi anotirani video posnetek. Skozi posnetek se osvežujejo anotacije, ki jih vmesnik pridobiva v vektorski obliki ločeno od videa. V obravnavanem primeru so anotacije

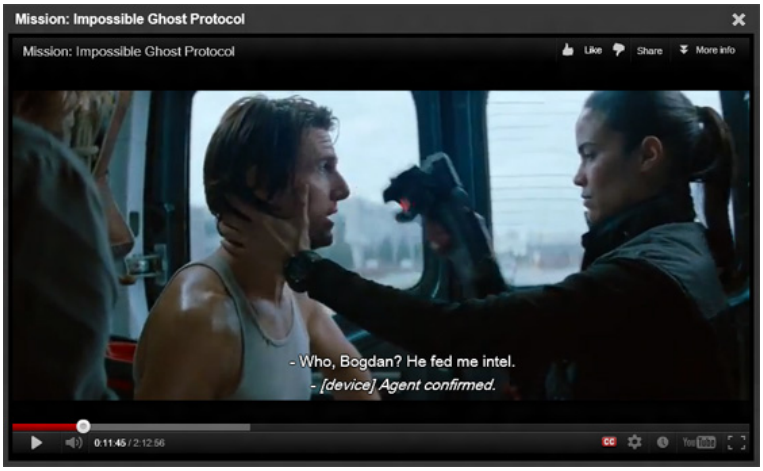
vizualizirani relzutati sledilnih algoritmov. Zahtevane tehnologije *HTML 5* in *Javascript* so najbolj primerne predvsem zato, ker so dostopne vsakomur in so prihodnost spletnega programiranja. Poznamo veliko dodatne programske opreme, ki preko grafičnih vmesnikov dodajajo anotacije video posnetkom. Vendar jih mora uporabnik prenesti in naložiti na računalnik. S to aplikacijo tega ne bo več potrebno, kajti zahtevano tehnologijo sedaj podpira veliko spletnih brskalnikov. Aplikacija nam omogoča predvajanje video posnetka z anotacijami, prikazovanje besedila v video posnetku ter interaktivnost preko miškinega kazalca. Najprej pa bomo opisali nekaj sorodnih primerov, ki jih lahko vidimo na spletu.

## 1.2 Sorodni pristopi

Na spletni strani `www.youtube.com` lahko naredimo praktično vse anotacije, ko urejamo naš video posnetek. Dodajamo lahko osnovne anotacije preko urejevalnika video posnetkov. Tam dodajamo tekstovne anotacije na video v oblačkih, hkrati pa lahko dodajamo tudi povezave na druge strani, slike in celo kratke inserte iz drugih youtube video posnetkov. Primere anotacij lahko vidimo na sliki 1.1.

Druga oblika anotacije so podnapisi, ki jih lahko dodajamo kot prepis v `.txt` obliki ali kot tekstovno datoteko z drugimi končnicami (primer: `.srt`). Tipičen izgled teh datotek je, da je poleg časa, ki označuje določen odsek v video posnetku, napisano besedilo, ki se bo pojavilo v tem času. Datoteke dodamo, ko urejamo video posnetek. Poleg tega moramo nastaviti tudi jezik podnapisov. Da ima video posnetek podnapise, opazimo takrat, ko se pod časovnico pojavi gumb *CC*. Po kliku na gumb upravljamo s podnapisi, kjer lahko nastavimo, v katerem jeziku se bodo pojavili. Če se podnapisi prikazujejo, se gumb obarva rdeče, drugače pa je obarvan v sivo. Izgled podnapisov lahko vidimo na sliki 1.2.

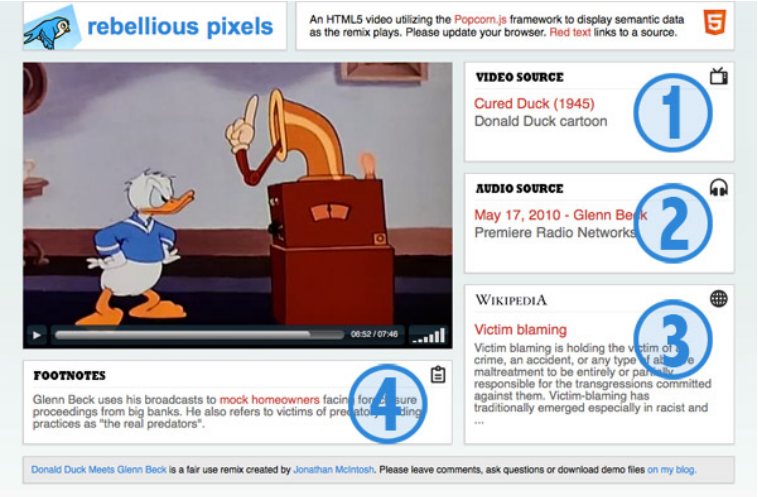

Slika 1.1: Izgled anotacij na spletni strani www.youtube.com




Slika 1.2: Izgled podnapisov na spletni strani www.youtube.com

Na spletu (vir [8]) imamo primer avtorja Jonathana McIntosha, ki nam prikazuje, kako predvajalnik deluje interaktivno z drugimi spletnimi viri. Predvajalnik predvaja kolaž video in avdio posnetkov. Med predvajanjem dodaja ali osvežuje nove podatke s pomočjo semantike in knjižnjice Popcorn.js. V ozadju je uporabnik določil spletne strani, iz katerih bo aplikacija prikazala podatke. Ob predvajanju video posnetka nam področje, ki ga označuje

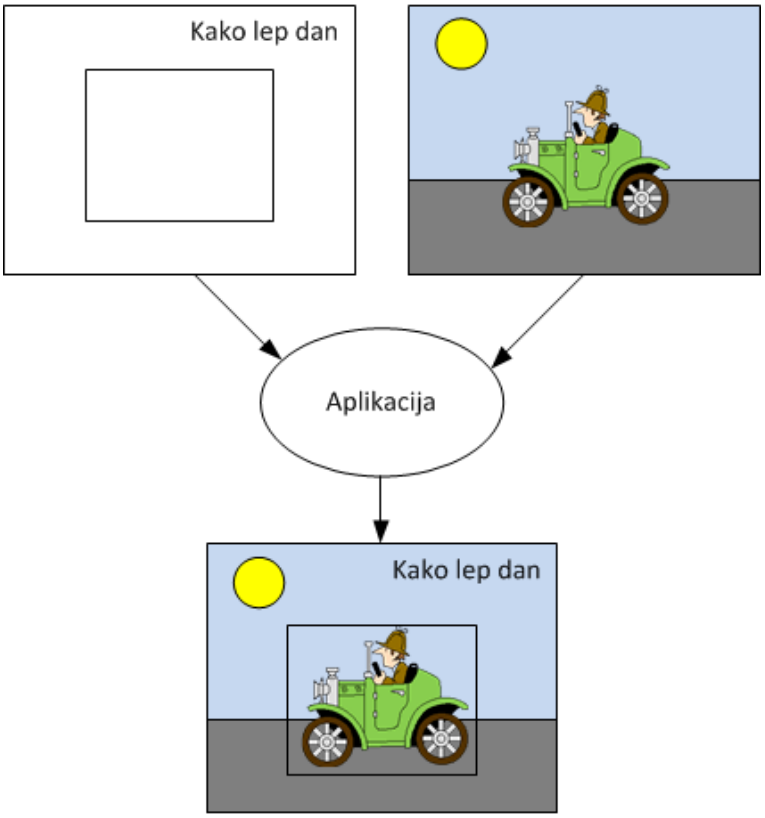
št. 1 na sliki 1.2, prikaže naslov trenutno predvajanega vizualnega posnetka v videu, področje št. 2 pa naslov trenutno predvajanega audio posnetka v videu. Področje št. 3 prikaže ustrezne podatke iz wikipedije, podorčje št. 4 pa dodatno razlago in pojasnila.




Slika 1.3: Pogled na semantični prikaz podatkov skupaj z video projekcijo

## 1.3 Osnovna ideja

Za razliko od zgornjih del se bomo osredotočili na obogateno prikazovanje anotacij. Podatke bomo pridobivali preko strežnika, kjer so shranjene datoteke z opisom anotacij. Preko vmesnika jih nato prikažemo, ko se predvaja video posnetek. Poznamo več oblik prikaza anotacij, npr. različni liki in besedilo. Zahtevano pa je tudi, da se prikaže interaktivnost preko premikanja miške po polju, na katerem bomo prikazovali anotacije. Bolj podrobno bomo postopke in prikaz opisali v naslednjih poglavjih, za vizualno predstavo si poglejte sliko 1.4.




Slika 1.4: Osnovna ideja aplikacije

## 1.4 Zgradba diplomske naloge

V poglavju 2 bomo opisali tehnologijo in orodja, s katerimi smo se srečevali. Prikazali smo tudi nekaj primerov uporabe tehnologij. V poglavju 3 bomo opisali izgled aplikacije in delovanje sistema, na katerem smo gradili projekt. V poglavju 4 bomo pokazali različne primere uporabe aplikacije in jih opisali. Za boljšo razlago so pri opisu dodane še slike.

# Poglavje 2

# Tehnologije in orodja

Naša aplikacija deluje preko spletnih tehnologij. Omejili smo se na uporabo HTML5 in Javascripta, znotraj Javascripta smo uporabljali knjižnjico JQuery. Podatke smo pridobivali od strežnika preko tehnologije AJAX in sicer preko datotečnega sistema JSON. Obliko aplikacije pa smo uredili preko slogovnega jezika CSS3. Da smo aplikacijo naredili, smo uporabljali orodje Adobe Dreamweaver CS5. V nadaljevanju poglavja bomo te tehnologije podrobneje opisali.

## 2.1 HTML5

HTML5 (Hyper Text Markup Language 5) je označevalni jezik za oblikovanje in predstavitev vsebine za svetovni splet. Kot izboljšava prejšnje verzije (HTML4) ima dodano podporo za najnovejšo multimedijo, hkrati pa je za ustvarjalce in programerje berljiv in lahko izvedljiv. Prinaša veliko novosti v sintaksi, med njimi tudi elemente `<video>` in `<canvas>`. Od predstavitve januarja 2008 se še vedno dopolnjuje in izboljšuje, podpira pa ga tudi že večina internetnih brskalnikov.

Naša spletna aplikacija sloni na dveh HTML5 elementih: `<canvas>` in `<video>`.

Predvajanje video posnetkov je z začetkom HTML5 protokolov postalo

lažje izvedljivo. Do takrat je bilo to možno preko vtičnika *flash*, po predstavitvi *video* elementa pa je le ta postal standard, ki ga podpira večina različnih video formatov. Ker pa vsi internetni brskalniki ne podpirajo vseh video formatov (glej tabelo 2.1), imamo možnost dodati več različnih formatov z več različnimi video elementi. Z atributi v elementu lahko sedaj v spletnem brskalniku lažje manipuliramo z video posnetki, ki so opisani v tabeli 2.2.

| brskalnik | H.264 | VP8 (WebM) | Ogg.Theora |
|---|---|---|---|
| Internet Explorer | Od verzije 9.0 | Ne podpira | Ne podpira |
| Mozila Firefox | Ne podpira | Od verzije 4.0 | Od verzije 3.5 |
| Google Chrome | Od verzije 3.0 | Od verzije 6.0 | Od verzije 3.0 |
| Apple Safari | Od verzije 3.1 | Ne podpira | Ne podpira |
| Opera | Ne podpira | Od verzije 10.60 | Od verzije 10.50 |

Tabela 2.1: Podpora brskalnikov za različne video formate v HTML 5 elementu `<video>` (vir [10] , stanje na dan 15.9.2012)

```
1 <video id="sampleMovie" width="640" height="360" preload
     controls>
2   <source src="HTML5Sample_H264.mov" />
3   <source src="HTML5Sample_Ogg.ogv" />
4   <source src="HTML5Sample_WebM.webm" />
5 </video>
```

Segment kode 2.1: Prikaz različnih video formatov v HTML 5

V primeru, da brskalnik ne podpira določenega video formata, si video posnetka ne moremo ogledati. To je bil od začetka uporabe elementa *video* pereč problem. Rešili so ga tako, da lahko sedaj v tem elementu kličemo več različnih video formatov naenkrat. Spletni brskalnik nato razbere pravilno končnico ter preko svojega predvajalnika prikaže video posnetek. Primer uporabe vidimo na primeru segmenta kode 2.1

| Atributi | Vrednost | Opis |
|---|---|---|
| *autoplay* | autoplay | Pove, ali se video predvaja od začetka takoj, ko bo pripravljen. |
| *controls* | controls | Pove, ali naj bo vidna kontrolna plošča (tipke za predvajanje, premor itd.). |
| *height* | pixels | Nastavi višino video predvajalnika. |
| *loop* | loop | Pove, ali se bo video, ko pride na konec, spet predvajal od začetka. |
| *muted* | muted | Pove, ali se bo video predvajal na tih način. |
| *poster* | url | Preko spletnega naslova določimo sliko, ki se pokaže, preden se video začne predvajati. |
| *preload* | auto, metadata, none | Pove, kako lahko nastavimo video prenos, ko se postavi spletna stran. |
| *src* | url | Nastavi spletni naslov video posnetka. |
| *width* | pixels | Nastavi širino video predvajalnika. |

Tabela 2.2: Atributi v HTML 5 elementu `<video>`

Atribute iz tabele 2.2 nastavljamo znotaj elementa *video*. V naši aplikaciji bomo uporabljali le nekatere izmed atributov. Nekaj atributov je določenih že avtomatsko, kot npr. atribut *controls*, ki nam pokaže ukazno vrstico video posnetka skupaj s časovnico. Poleg atributa *controls* se nam samodejno

nastavi še atribut *preload*, pri katerem je potrebno po začetku predvajanja počakati, da se prenese določen procent videa. Potem predvajalnik začne predvajati video posnetek. Preko klica atributov *height* in *width* smo kasneje lahko določili velikost grafičnega elementa *canvas*, z atributom *src* pa smo nastavili pot, preko katere je video posnetek dosegljiv.

Tako kot video element se je s predstavitvijo HTML5 pojavil nov grafični element `<canvas>`, ki omogoča risanje dvo- ali tridimenzionalnih vizualnih objektov na spletno stran. Tega prej ni bilo mogoče narediti, razen preko vtičnika *flash* ali javanskih programov. Da pa lahko rišemo na več plasteh nam omogoča skripta *Javascript*. Grafični element `canvas` sedaj podpirajo vsi novejši spletni brskalniki, kar je opisano v tabeli 2.3.

| brskalniki | podpora grafičnega elementa `<canvas>` |
|---|---|
| `Internet Explorer` | Od verzije 9 naprej |
| `Mozila Firefox` | Od verzije 12 naprej |
| `Google Chrome` | Od verzije 20 naprej |
| `Apple Safari` | Od verzije 5.1 naprej |
| `Opera` | Od verzije 12 naprej |

Tabela 2.3: Podpora brskalnikov grafičnega elementa `<canvas>`

Ko definiramo grafični element canvas na spletni strani (segment kode 2.2), je samodejno prednastavljen tako, da je neviden. Vendar je vedno postavljen v širini in višini, ki ju določimo znotraj elementa. Znotraj elementa določimo še atribut *id*, ki nam je v pomoč pri klicih grafičnega elementa *canvas*, če ga je potrebno oblikovno spremeniti ali pa določiti funkcionalnost preko atributov.

```
1 <canvas id="myCanvas" width="200" height="100"></canvas>
```

Segment kode 2.2: Primer uporabe grafičnega elementa canvas

V grafičnem elementu canvas lahko rišemo različne vektorske oblike, od osnovnih do bolj kompleksnih. Pri dvodimenzionalnem elementu imamo koordinati x in y, preko katerih pozicioniramo, kje bomo risali določene ele-

mente. Tabela 2.4 prikazuje metode, ki delujejo v dvodimenzionalnem prostoru. Z njimi določimo, kaj in kako bomo narisali v naš grafični element canvas.

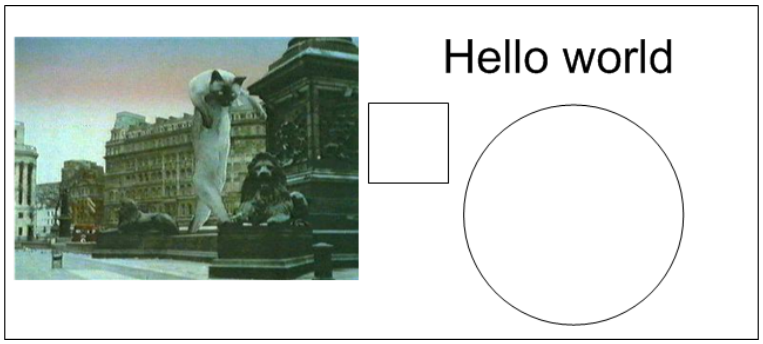
Na sliki 2.1 vidimo rezultat, ki jo ustvari segment kode 2.3. V tem segmentu se kličejo funkcije, ki rišejo like. Pri funkciji `drawImage()` vidimo, da nariše za določene koordinate sliko. Funkcija `fillText()` na določene koordinate nariše besedilo, font in velikost pa sta nastavljena preko funkcije `font()`. Funkcija `arc()` nastavi koordinate in pot za krog brez polnila. Risanje kroga se začne s funkcijo `beginPath()`, konča pa s funkcijo `stroke()`, ki krog nariše. Funkcija `strokeRect()` pa nariše pravokotnik, ki pa nima barvnega polnila, ampak samo obrobi pravokotnik s črto. Te metode se izvajajo po vrstnem redu, kot so napisane v segmentu kode 2.3.

```
1  <body>
2  <img id="scream" src="img_the_scream.jpg" alt="The Scream"
       width="220" height="277">
3  <canvas id="myCanvas" width="650" height="300" style="border
       :1px solid #d3d3d3;"></canvas>
4  <script>
5  var c=document.getElementById("myCanvas");
6  var ctx=c.getContext("2d");
7  var img=document.getElementById("scream");
8
9  ctx.drawImage(img,10,10);
10 ctx.font="30px Arial"; ctx.fillText("Hello World",250,50);
11 ctx.strokeRect(250,100,50,50);
12 ctx.beginPath();
13 ctx.arc(400,200,90,0,2*Math.PI);
14 ctx.stroke();
15 </script>
16 </body>
17 }
```

Segment kode 2.3: Uporaba metod v grafičnem elementu *canvas* (vir [11])

| Metoda | Parametri | Opis |
|---|---|---|
| `strokeStyle()` | Barva črte v HEX vrednosti, RGB ali pa določena kot niz | Metoda določi barvo črte, ki se bo risala. |
| `fillStyle()` | Barva črte v HEX vrednosti, RGB ali pa določena kot string | Metoda določi barvo polnila lika, ki se bo risal. |
| `strokeRect()` | x1, y1, x2, y2 | Metoda določi točki levo zgoraj in desno spodaj in nariše pravokotnik brez polnila. |
| `fillRect()` | x1, y1, x2, y2 | Metoda določi točki levo zgoraj in desno spodaj in nariše pravokotnik, katerega napolni z barvo. |
| `strokeLine()` | x1, y1, x2, y2 | Metoda določi točki levo zgoraj in desno spodaj ter nariše črto. |
| `beginPath()` | / | Metoda določi začetek risanja. |
| `arc()` | x1, y1, r, alpha, dolžina krožnice, clockwise (true or false) | Metoda določi pogoje za risanje. |
| `scale()` | Razmerje, velikost razmerja | Metoda določi razmerje med širino in višino. |
| `stroke()` | / | Metoda nariše črto. |
| `font` | string | Metoda izbere družino fonta in velikost v pikslih. |
| `strokeText()` | text,x1,y1 | Metoda nariše tekst, ki je lociran s koordinatama in nima polnila. |
| `fillTect()` | text,x1,y1 | Metoda nariše tekst, ki je lociran s koordinatama in ima določeno barvno polnilo. |
| `drawImage()` | image,x1,y1 | Metoda nariše sliko, ki je locirana s koordinatama. |

Tabela 2.4: Metode v grafičnem elementu `<canvas>`




Slika 2.1: Izgled segmenta kode 2.3 na grafičnem elementu *canvas*

## 2.2 CSS3

Za oblikovanje spletne strani smo uporabili *CSS3* (Cascading Style Sheets 3), ki je slogovno oblikovni jezik. Uporablja se za opis dokumenta (videz in format) zapisanega v označevalnem jeziku. Najpogosteje se ta jezik uporablja za spletne strani, napisane v označevalnem jeziku HTML. Najnovejša verzija CSS3 je napisana v skladu z najnovejšimi pridobitvami tehnologije HTML5. Z novimi tehnologijami so prišli tudi najnovejši slogovni in oblikovni stili, povečala pa se je tudi podpora za različne spletne brskalnike, predvsem za Opero (pri navajanju stilov dodamo predpono `-o-`), Mozilo Firefox (pri navajanju stilov dodamo predpono `-moz-` ) ter Apple Safari in Google Chrome (pri navajanju stilov dodamo predpono `-webkit-` ). Pri spletni aplikaciji smo si z njim pomagali, da smo elemente strani lahko postavili, nekatere pa tudi oblikovno izboljšali, da so bolj vidni.

## 2.3 Javascript in JQuery

Aplikacija potrebuje spletni programski jezik, ki bi deloval v ozadju. Za to je najbolj primeren *Javascript*, ki se uporablja za ustvarjanje interaktivnih spletnih strani. Da pa lahko interaktivno spreminjamo spletno stran, imamo z Javascriptom to možnost, da spreminjamo ali pa dodajamo elemente *HTML*

in *CSS*. *JQuery* je *Javascript* ogrodje ali knjižnica, ki vsebuje vnaprej napisane funkcije iz jezika Javascript. Filozofija ogrodja *JQuery* se glasi "Write less, do more"(Z manj pisanja naredi več), kar pomeni da je programerjem z uporabo te knjižnjice potrebno napisati bistveno manj, da dosežejo iste rezultate. Preko tega programskega jezika smo v ozadju sledili video posnetku, spreminjali obliko anotacij, dodajali funkcionalnost aplikacije ter hkrati tudi dinamično spreminjali izgled aplikacije. Primere uporabe so prikazani v poglavju 3.

## 2.4 JQuery AJAX

V ozadju aplikacije potrebujemo tudi funkcije, ki bi se povezale s strežnikom in iz njega pridobile datoteke, s tem pa podatke, ki jih potebujemo za risanje anotacij. Uporabili smo *ajax()* funkcijo (Asynchronus JavaScript and XML) v *JQuery* knjižnjici. Funkcija *ajax()* izvaja asinhrone HTTP zahteve in se zanaša na strežnik, da posreduje odgovore za pridobljene podatke. V klicu funkcije imamo na voljo veliko različnih nastavitev. Obdelave podatkov je mogoče doseči z uporabo nastavitve *datatype*. Poleg navadnega XML so ostali tipi podatkov HTML, JSON, JSONP ali `.txt` besedilo. Če je tip JSON, se podatki razčlenijo kot objekti, tako tudi lahko dostopamo do njih. Primer uporabe je bolj podrobno opisan v poglavju 3.

## 2.5 JSON

Za aplikacijo morajo biti datoteke iz strežnika enostavne in majhne. Zato je za našo aplikacijo primeren dokument formata JSON (JavaScript Object Notation). Je preprost format za izmenjavo podatkov, ki temelji na objektnem programiranju v javanski tehnologiji. Je enostaven za generiranje podatkov ter za spletne vmesnike, ki pridobivajo podatke preko njega. Primer uporabe je prikazan v segmentu kode 2.4.

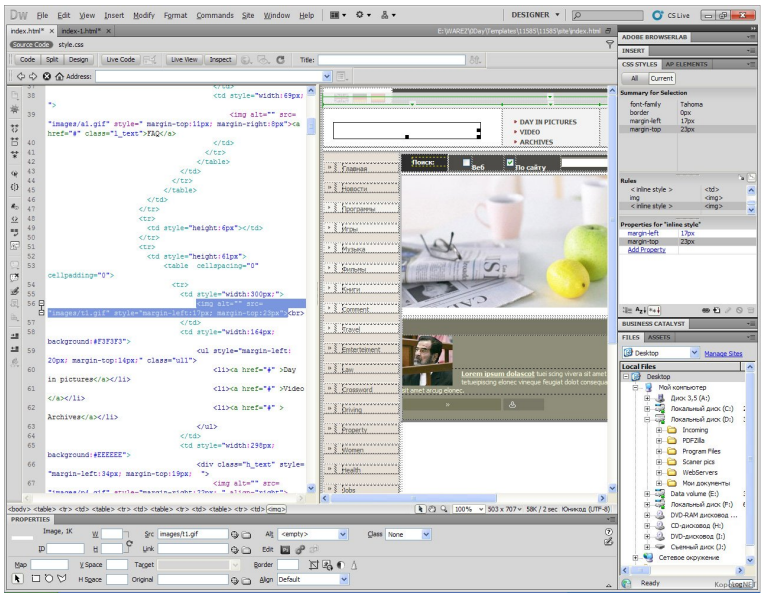
```
1 {
```

```
2     "firstName": "John",
3     "lastName": "Smith",
4     "age": 25,
5     "address": {
6         "streetAddress": "21 2nd Street",
7         "city": "New York",
8         "state": "NY",
9         "postalCode": "10021"
10    },
11    "phoneNumber": [
12        {
13            "type": "home",
14            "number": "212 555-1234"
15        },
16        {
17            "type": "fax",
18            "number": "646 555-4567"
19        }
20    ]
21 }
```

Segment kode 2.4: Primer uporabe JSON (vir [5])

## 2.6 Adobe Dreamweaver CS5

Pri našem delu smo mogli uporabiti orodje, preko katerega bi lahko sprogramirali in preizkušali aplikacijo. Zaradi dobrih lastnosti smo se odločili, da bomo uporabljali Adobe Dreamweaver iz kolekcije CS5 (Creative Suite 5). Je vizualni urejevalnik za ustvarjanje in upravljanje spletnih mest ter strani. Dreamweaver ponuja napredna orodja za načrtovanje in postavitev, kot npr. uporaba dinamičnih HTML funkcij (animirane plasti) brez pisanja kode. Poleg tega ponuja tudi pregled kode, objavljanje napak, ogled postavitve in delovanja na spletni strani.



Slika 2.2: Pogled na delovno okolje Adobe Dreamweaver CS5

# Poglavje 3

# Programska rešitev aplikacije

## 3.1 Izgled aplikacije

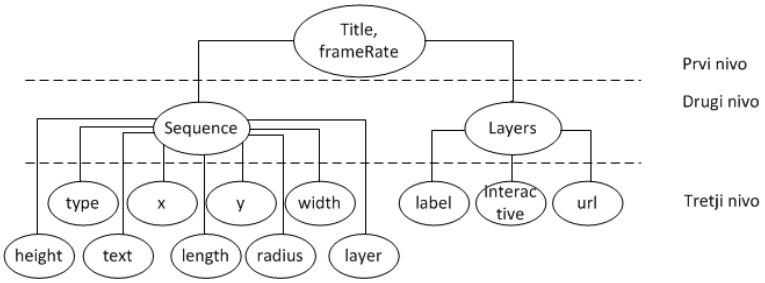
V laboratoriju za umetne vizualne spoznavne sisteme smo sestavili sistem, ki sledi določenim objektom v video posnetku. Sistem shrani podatke v datoteko, do katere dostopamo preko strežnika. Zahtevano je, da se aplikacija naredi v tehnologijah HTML5 in Javascript, ki omogočata podatke iz strežnika preoblikovati v anotacije. Zahteva pride v poštev predvsem zato, ker bomo tako lahko aplikacijo upravljali preko spleta. Spletni uporabniški vmesnik ima na voljo to možnost, da po kliku na gumb `Play` prikaže različne vrste vizualizacij, ki bi se prikazovale po sledeh poslanih koordinat skozi cel video posnetek ali le del video posnetka. Ob tem bi uporabnik imel možnost spreminjati vidnost vizualizacij – se pravi bo lahko sledil enemu samemu objektu ali pa večim, odvisno od izbire. Poleg tega bo uporabnik lahko lokalno shranil tekst v obliki dokumenta z anotacijami. Preko spremljanja narisanega lika uporabnik lahko tudi klikne na povezavo, ki se pojavi na canvasu.


Slika 3.1: Pogled z nivoji

Predvajalnik ima časovno premico, po kateri se lahko uporabnik premika in prestavi video na določeno časovno komponento. Časovna komponenta se šteje v časovni konstanti `fps` (Frames per Second) in je tudi prikazana poleg časovnice, da se uporabnik lahko preko nje navigira. Ima tudi dva gumba (»Play/Pause« in »Stop«), s katerima lahko uporabnik opravlja z videom. Nad gumboma pa se na grafičnem elementu canvasu izriše predvajani video z anotacijami. Ob strani se izrišejo gumbi, s katerimi lahko določamo vidnost plasti, na katerih se rišejo liki. Poleg gumbov se pojavijo še tekstovna polja za lokalno shranjevanje teksta v obliki dokumenta z anotacijami.

## 3.2 Izvedba aplikacije

Zahtevano je, da si bomo podatke pošiljali preko spleta z datoteko, ki vsebuje anotacije. Zaradi enostavnosti in hitrega dosega podatkov smo se odločili, da privzamemo strukturo dokumenta JSON. Strukturo datoteke je bilo potrebno karseda poenostaviti, da zaradi velikosti ne bo predolgega sprejemanja podatkov. V prvem nivoju je po zahtevi naslov dokumenta (title) ter koliko slik na sekundo (fps) bo video predvajal (frameRate), kot kaže slika 3.2.




Slika 3.2: Hierarhija dokumenta z anotacijami

```
"title":"Results for sequence Bicycle",
"frameRate": 10
```

Segment kode 3.1: Osnovni podatki

V tabeli objektov plasti (layers) so napisane oznake plasti (label). Interaktivnost plasti je določena preko objekta "interactive". Če je vrednost v objektu "false", s to plastjo ne naredimo nič (razen prikazovanja anotacij). Če pa je vrednost v objektu "true", na plast dodamo spletni naslov. To pomeni, da bo ob prikazovanju anotacij možnost preklapljanja na spletno stran. Več o tem je razloženo v poglavju 4. Skupno število oznak pa tudi pove, koliko jih bo na spletni strani (glej segment kode 3.2).

```
"layers":
[
  {"label":"lgtmatlab", "interactive":"true", "url":"http:\\
     www.rtvslo.si" },
  .
  .
  .
]
```

Segment kode 3.2: Poimenovanje plasti

V tabeli zaporedij slik oziroma sličic (sequence) so objekti, ki določajo tip lika (type), koordinate lika (za različne like različna poimenovanja koordinat) ter oznako plasti (layer). Število slik v sekvenci pove, kolikšna je dolžina videa. Če je anotacija besedilo, imamo poleg koordinat še dolžino predvajanja

(length) ter besedilo, ki se bo pojavilo (text). Primer te kode lahko vidimo na segmentu kode 3.3.

```
"sequence":
[
  [{"type":"rect","x":154,"y":94,"width":18,"height":48,"
     layer":"lgtmatlab"},
   {"type":"circle","x":290,"y":50,"radius":30,"layer":"
      semaphor"},
   {"type": "text", "x":50, "y": 50, "length":100, "text":"
      Lep pozdrav vsem"}],
  .
  .
  .
]
```

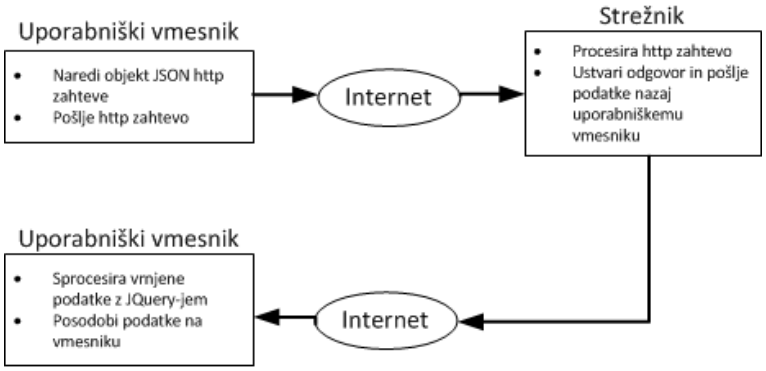
Segment kode 3.3: Tabela sekvenc

Ko imamo na voljo dokument z anotacijami, lahko preko funkcije `.ajax()` pridobivamo podake iz strežnika preko spleta. Funkcija je na voljo v programski kodi JQuery. Poleg te funkcije poznamo tudi funkcijo `.getJSON()`. Pri tej funkciji nimamo toliko nastavitev kot pri funkciji `.ajax()`, saj večino dela opravi samodejno. Ker ne bomo pošiljali podatkov nazaj, ampak jih bomo le sprejemali, smo uporabili HTTP metodo `GET` sprejema podatkov. Poleg tega je potrebno nastaviti tako, da bo sprejemal tip datoteke JSON ter url te datoteke. To pošljemo preko HTTP zahteve. Če vse poteka brez napak, nadaljujemo z delom preko nastavitve success. Le ta preveri, če je strežnikov odziv preko HTTP pozitiven, če ni, pa ne naredi ničesar oziroma vrne napako.

```
$.ajax({
    type: "GET",
    dataType: "json",
    url:"https://dl.dropbox.com/u/95398407/
       JansaNejcDiplomskaNaloga/example.json",
    success: function(data){
    // nadaljujemo s polnjenjem tabel
    }
```

```
8 });
```

Segment kode 3.4: Klicanje funkcije `.ajax()`




Slika 3.3: Graf prikaza delovanja funkcije `ajax()`

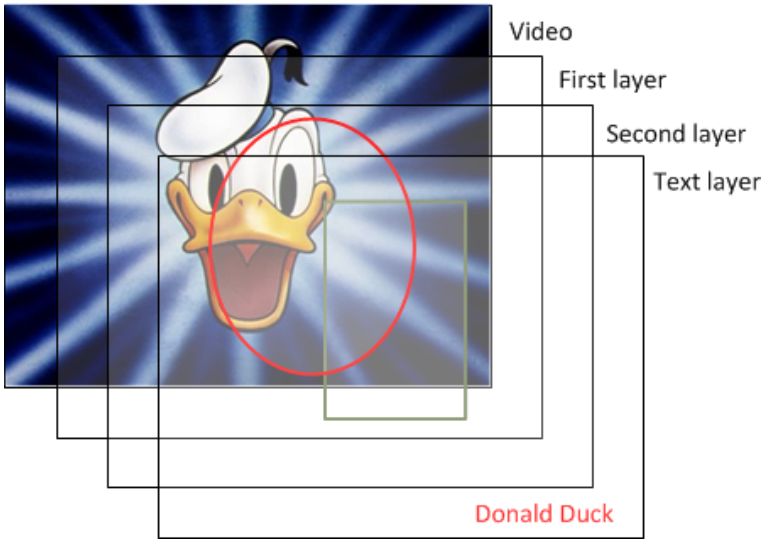
Prvi problemi so se pojavili pri testiranju aplikacije. Ko smo hoteli aplikacijo stestirati lokalno, nam je funkcija `ajax()` vrnila nazaj napako, ker ni mogla sprejeti datoteke. Problem je bil v spletnem brskalniku zaradi navzkrižnih komunikacij med domenami. Če datoteke ni na domeni spletne strani, je spletni brskalnik ne more prebrati ali pa vrne napako. Tako smo imeli dve možnosti: prva možnost je, da si lokalno nastavimo spletni strežnik, druga možnost pa je, da ta dokument gostuje na drugi spletni domeni oz. na drugem strežniku, kar smo tudi storili.

Preko nastavitve success kličemo nato funkcijo s parametrom, kjer se shranijo vsi podatki, ki jih dokument z anotacijami vsebuje (glej segment kode 3.4). Znotraj funkcije shranjujemo v globalne spremenljivke podatke iz dokumenta z anotacijami. Poleg shranjevanja podatkov še generiramo grafične elemente canvas (glej segment kode 3.5). Liki se shranjujejo v objekt, ki vsebuje toliko tabel, kolikor imamo na voljo likov. Določili smo, da lahko na plasti narišemo kvadrat, krog, elipso, črto in točko. Poleg likov v tabele shranjujemo tudi besedilo.

```
1 <canvas id="lgtmatlab" class="show" style="position:absolute;
      z-index:2;" width="320" height="240"></canvas>
```

Segment kode 3.5: Generiranje grafičnih elementov *canvas*

Nato je potrebno dane podatke iz tabel narisati v grafičnem elementu *canvas*. Še prej pa moramo video prikazati na eni izmed plasti in ne v predvajalniku, ker drugače ne bi mogli spreminjat videa skupaj z grafičnimi elementi *canvas*. Razlogi za to so v tem, da na element *video* ne moremo risati anotacij, kar pa lahko naredimo na grafičnem elementu *canvas*. Video predvajalnik je bilo potrebno skriti, pokazati pa plast, na katerem se projecira video. K temu je bilo potrebno še dodati gumbe za upravljanje z videom, kot sta gumba »`Play/Pause`« in gumb »`Stop`«. Koncept risanja večih likov na plasteh je tak, da ima vsak lik svojo plast. Med seboj pa so plasti razdeljene po z- koordinati (tretja dimenzija), s tem da ima video plast najnižjo koordinato. Kako so plasti urejene, vidite na sliki 3.4.




Slika 3.4: Pogled na plasti

Video plast je narejena po naslednjem konceptu: Iz videa, ki se predvaja na video predvajalniku in ki smo ga dali na spletno stran preko elementa *video*, se bo preko risalnih funkcij projeciral na plast. Najprej je potrebno

preračunati dolžino in širino videa, da se lahko določi velikost plasti. Nato je potrebno za vsak okvir iz videa narediti posnetek in ga projecirati na plast. To funkcijo moramo ponavljati ob času `1000ms/fps (frames per second)`, podatek za čas fps pa dobimo v dokumentu z anotacijami. Ponavlja se toliko časa, dokler se video ne odvrti do konca oziroma do takrat, ko uporabnik klikne na gumb »`Pause`« ali »`Stop`«. Funkcije, ki kličejo risanje na plasti, so prikazane na segmentu kode 3.6.

```
1  theVideo.addEventListener('play', function() {
2        setInterval(snap, 1000/points.frameRate);
3        .
4        .
5        .
6     }, false);
7
8  theVideo.addEventListener('paused', function() {
9        clearInterval();
10 }, false);
11
12 theVideo.addEventListener('ended', function() {
13       clearInterval();
14    }, false);
```

Segment kode 3.6: Prikaz dogajanje med upravljanjem z videom

Problem nastane, ko slike preslikavamo eno čez drugo in pri tem ne nastane video ampak skupek slik, nametanih eno na drugo. Med vsakim novim klicem funkcije, ki projecira posnetek na plast, je potrebno klicati še funkcijo, ki izbriše na plasti tisto, kar je bilo prej projecirano. Ta funkcija je prikazana na segmentu kode 3.7.

```
1  var blank = function(i) {
2        ctx[i].clearRect(0,0,ctx[i].canvas.width, ctx[i].canvas
            .height
3  };
```

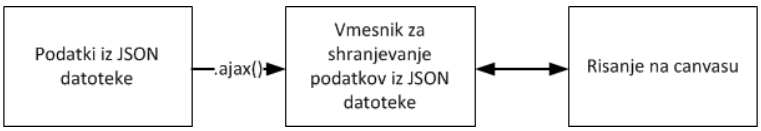
Segment kode 3.7: Funkcija `blank()`

Da lahko rišemo na plasti, je potrebno najprej določiti, v kateri dimenziji se bo risalo. Iz HTML elementa dobimo informacije o dolžini in širini, z metodo `getContext` (glej segment kode 3.8) pa določimo spremenljivki dimenzijo, preko katere lahko delamo z metodami iz tabele 2.4.

```
1 theCanvas[i] = document.getElementById('Layer1');
2 ctx[i] = theCanvas[i].getContext('2d');
```

Segment kode 3.8: Lastnosti grafičnega elemnta canvasa se shranijo v spremenljivke

Risanje likov na plasti določimo s tem, ko preko shranjenih podatkov iz datoteke z anotacijami ugotovimo, kateri lik je potrebno narisati. Poleg te ugotovitve je potrebno tudi ugotoviti še plast, na katero se bo risalo. Ko imamo vse potrebne podatke, s funkcijami iz tabele 2.4 narišemo like na plasti. Po pritisku na `Play` se v času enega okvirja narišejo na plasteh liki, ki so shranjeni v tabelah objekta. Funkcija `blank()` pa za naslednji okvir izbriše vse, kar je bilo v prejšnjem narisano. Tako se poleg videa narišejo tudi vse anotacije na plasteh ki so določene.




Slika 3.5: Graf prikaza delovanja vmesnika

V datoteki z anotacijami pa obstaja primer, ko želimo za določeno dolžino videa prikazati besedilo kot eno izmed anotacij. Ker se po videu besedilo ne premika, ni potrebe, da bi bila zabeležena v dokumentu z anotacijami v tabeli sličic pod vsak okvir. Pomembno je, da je zabeležena v začetnem okvirju, ker kasneje z objektom *length* pridobimo potrebno dolžino trajanja. Besedilo, ki je shranjeno v objektu *text*, kasneje prikažemo na plasti, ki smo jo generirali, ko smo dobili dokument z anotacijami. Vse to je prikazano v poglavju 4.

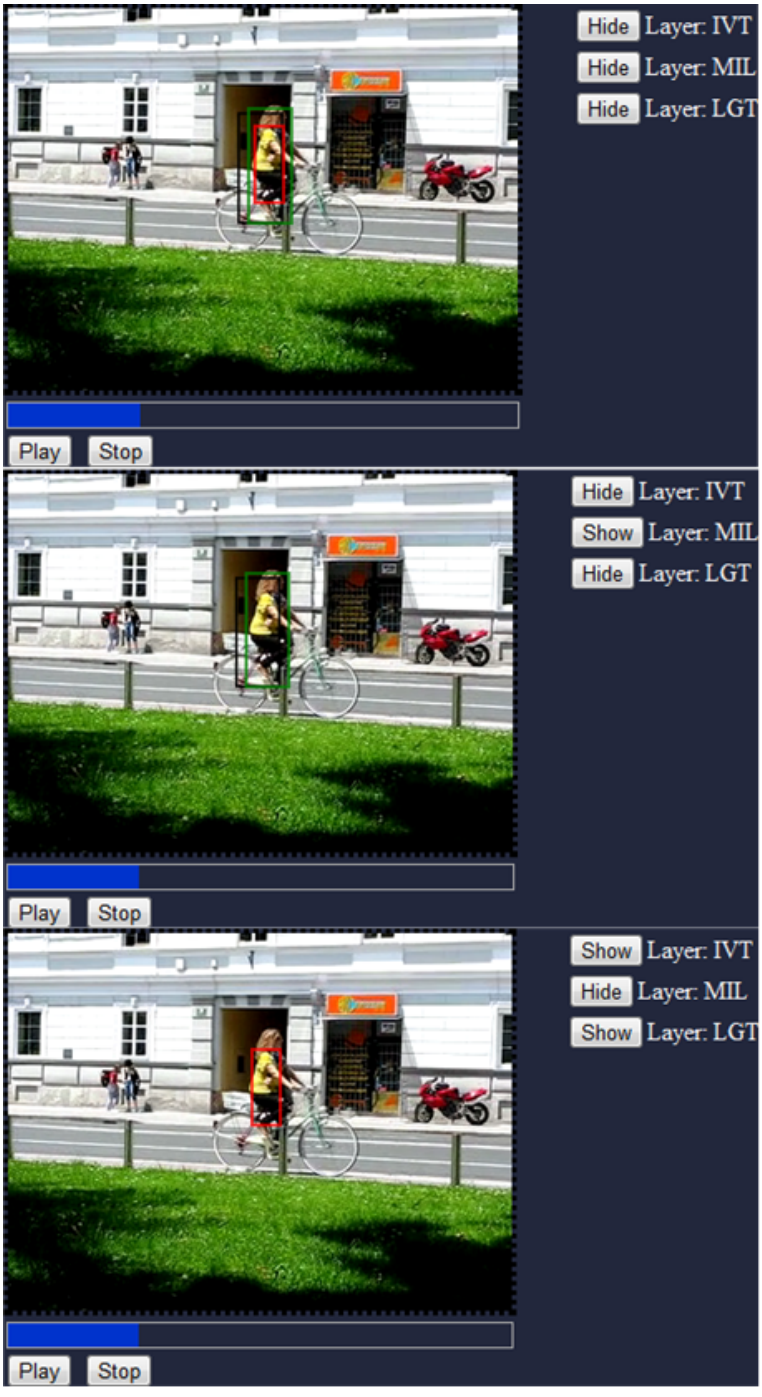
# Poglavje 4

# Primeri uporabe

## 4.1 Predvajanje video posnetka z anotacijami

Uporabniški vmesnik se s pritiskom na gumb `Play` aktivira in začne predvajati video. Na video predvajalniku se pokaže video, na katerem opazimo anotacije. Po začetku predvajanja video posnetka lahko uporabnik vidi, da so to določeni liki, ki so narisani na plasteh. To so anotacije, preko katerih lahko uporabnik sledi določenemu objektu. Ko uporabnik pritisne tipko `Pause`, se video ustavi, s tem pa tudi anotacija. Če hoče uporabnik videti anotacijo kje drugje na časovni premici, lahko preko nje določi časovno komponento posnetka in od tam naprej se bo ta tudi vrtel, z njim pa anotacije. Po pritisku na tipko `Stop` se video postavi na začetek. Da uporabnik ve točno, kje se z vrtenjem video nahaja, ima ob strani za pomoč stanje na časovni premici, ki se šteje v časovni konstanti `fps`. Sledenje anotacijam je prikazano na sliki 4.1.


Slika 4.1: Prikaz sledenja anotacije na kolesarki




Slika 4.2: Po kliku na gumb se anotacija skrije ali prikaže

Med predvajanjem ima uporabnik na voljo gumb, preko katerega lahko skrijemo ali pokažemo plast, na katerem je narisana anotacija. Teh gumbov je lahko več, odvisno od tega, koliko anotacij imamo med predavanjem. Da spoznamo, za katero anotacijo gre, imamo ob gumbu napisano ime plasti. Napis v gumbu se s spreminjanjem vidnosti anotacije spremeni, na videu pa po kliku na gumb uporabnik lahko opazi, da se anotacija skrije ali pojavi, kar je pokazano na sliki 4.2. Skrivanje anotacij pride prav takrat, ko hočemo prikazati specifično anotacijo, da se lahko na njo osredotočimo.

## 4.2 Prikazovanje besedila v videu in shranjevanje besedila

Hkrati z anotacijami, ki so v obliki likov, lahko vidimo anotacije, ki nam prikažejo besedilo. Te se pojavljajo takrat, kot je to določeno v datoteki z anotacijami. Uporabnik pride do teh anotacij tako, kot je opisano v zgornjem primeru, primer pa vidimo na sliki 4.3.

Poleg sledenja lahko shranjujemo v dokument še tekstovno plast z naslednjimi parametri:

- tekst, ki se bo prikazal na plasti,
- X in Y koordinati za prikaz na plasti,
- Začetni in končni okvir za trajanje teksta na plasti.

Ob predvajalniku imamo tekstovna polja, kjer vnašamo potrebne podatke (slika 4.4). V prvo tekstovno polje `Text input` vnesemo poljubni tekst, ki se bo ob uporabi lokalnega dokumenta z anotacijami prikazal. V drugo tekstovno polje `Start Frame` vnesemo začetni okvir. V tretje tekstovno polje `End Frame` vnesemo končni okvir, do katerega se bo prikazal tekst. V četrto tekstovno polje `X coordinate` vnesemo `x` koordinato na plasti. V peto tekstovno polje `Y coordinate` vnesemo `y` koordinato na plasti.


Slika 4.3: Prikaz anotacije z besedilomi

Po kliku na gumb »Submit« uporabnika v primeru pravilnega vnosa opozori, da je bilo vse pravilno. Če je uporabnik vnesel napačne podatke oz. če je kakšen podatek pozabil vpisati, se ob tekstovnem polju pojavi ali pojavijo opozorila. Če je z vnašanjem podatkov bilo vse v redu, se uporabniku lokalno shrani datoteka, ki ima iste lastnosti kot datoteka z anotacijami, ki jo dobimo preko strežnika.


Slika 4.4: Postopek dodajanja besedila na plasti

## 4.3 Interaktivnost aplikacije


Slika 4.5: Prikaz nove anotacije, ko gremo z miškinim kurzorjem čez drugo anotacijo

Medtem ko se predvaja video, na plasti uporabnik opazi anotacijo, ki sledi nekemu objektu (slika 4.5). Če gre čez njo z miško, se mu na spodnjem delu plasti pojavi rdeči pravokotnik, na katerega lahko uporabnik klikne. Če pa imamo anotacijo, ki se ne premika, lahko uporabnik klikne kar direktno na

njo. Po kliku ga prestavi na novo spletno stran v novem oknu ali zavihku, odvisno v katerem spletnem brskalniku se nahaja. Spletno stran pridobimo iz dokumenta z anotacijami, tako da uporabnik na spremembo spletne strani ne more direktno vplivati.

# Poglavje 5

# Zaključek

Cilj diplomske naloge je bil narediti aplikacijo, ki bo prikazala anotacije na videu, poleg tega pa bo obogatila prikaz z interaktivnostjo. V diplomskem delu smo predstavili pojem video anotacije. Analizirali smo vrste video anotacij in njihovo uporabo. Osredotočili smo se na vizualno prezentacijo video anotacij v predvajalniku. Raziskali smo uporabo spletnih tehnologij za prikazovanje anotacij v grafičnem elementu canvas. Prednost tega je, da je sedaj prikazovanje preko HTML5 tehnologije vidno vsakomur na spletu. HTML5 tehnologije so skupaj z Javascript in JQuery knjižnjicami prihodnost spleta, saj se še vedno dopoljnujejo in razvijajo. Naredili smo prikaz anotacij na osnovi sledilnih algoritmov, kar pomeni, da anotacije sledijo določenemu objektu na video predvajalniku. Dodali pa smo še nekaj vmesnikov, da lahko uporabnik bolje izkoristi anotacije in jih obogati s tekstom. Prikazali pa smo tudi, da je možno anotacijam slediti po plasti z miškinim kazalcem in preko tega dodati novo funkcionalnost na plasti.

Vmesnik z uporabo funkcjie `ajax()`, ki pobira podatke iz datoteke z anotacijami, lahko trenutno dodaja podatke samo iz ene datoteke. Aplikacijo bi lahko izboljšali s tem, da bi ta vmesnik združil več različnih datotek z anotacijami v eno in kasneje iz nje črpal potrebne podatke. Pri tem bi se, ne le samo povečala uporabnost, ampak bi tudi zmanjšali velikost vseh datotek, če bi dodali vse podatke v eno samo. Uporabnik bi to lahko videl s tem, da

bi se mu pojavilo več anotacij, imel bi več možnosti prikaza ter selekcije med njimi.

Obogaten prikaz video anotacij bi lahko izboljšali tudi z uporabo dodatnih knjižnjic. Ena izmed njih je Popcorn.js. Preko te knjižnjice bi lahko uporabnik na video časovnici dodajal preko semantičnega spleta razlage, slike in ostalo multimedijo, kar bi zelo pripomoglo npr. pri učenju. S tem bi bilo možno video posnetek združiti z učno snovjo, lahko tudi preko video anotacij. Študentje bi imeli s tem na voljo efektiven vizualni kot tudi slušni način učenja.

Vmesniki, ki so v tem konceptu predstavljeni, bi se lahko izboljšali na nekaj načinov. Možnost izboljšave je spreminjanje barv, debeline črte in motnost polnila barve anotacij. Razvita aplikacija vsekakor to omogoča, vendar na način, na katerega uporabnik ne more vplivati. Tako bi za to bilo potrebno dodati nov vmesnik, preko katerega bi uporabnik lahko izbiral določeno barvo na barvnih lestvicah, nastavil pa bi lahko tudi polnjenje barve, motnost, če pa tega ne bi želel spreminjati, pa bi lahko spremenil še debelino črte.

Implementirana aplikacija je pripravljena za uporabo v praksi. Sam koncept diplomske naloge je bil uspešno zastavljen. Pokazalo se je, da se z modernimi spletnimi tehnologijami da narediti koristno aplikacijo z uporabniku prijaznimi vmesniki. Namen, da bi z anotacijami obogatili prikaz videa v tehnologiji HTML 5 je torej uspel.

# Literatura


[1] (2012), HTML 5, dostopno na:
http://en.wikipedia.org/wiki/HTML5

[2] (2012), CSS3, dostopno na:
http://sl.wikipedia.org/wiki/CSS

[3] (2012), JQuery, dostopno na:
http: //im.scv.si/wiki/index.php/Jquery

[4] (2012), Javascript, dostopno na:
http://sl.wikipedia.org/wiki/JavaScript

[5] (2012), JSON, dostopno na:
http://en.wikipedia.org/wiki/JSON

[6] (2012), Youtube annotations, dostopno na:
http://googlesystem.blogspot.com/2008/06/youtube-annotations.html

[7] (2012), Adobe Dreamweaver CS5, dostopno na:
http://corp.kaltura.com/Products/Features/Video-Editing-Tools

[8] (2012), Semantic Web, dostopno na:
http://www.rebelliouspixels.com/semanticremix/

[9] (2012), HTTP, dostopno na:
http://sl.wikipedia.org/wiki/HTTP

[10] (2012), HTML5 element video, dostopno na:
http://www.w3schools.com/html/html5_video.asp

[11] (2012), HTML5 grafični element canvas - primer, dostopno na:
http://www.tutorialspoint.com/html5/canvas_drawing_rectangles.htm

[12] P. Lubbers, B. Albers, F. Salim. "Using the HTML5 Canvas API", v: Pro HTML5 Programming (Apress), 2010, str. 25–64.