Multimodale Steuerung von HTML Präsentationen

Elisabeth Baumgartner

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juni 2014

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 30. Juni 2014

Elisabeth Baumgartner

Inhaltsverzeichnis

Kurzfassung

In den letzten Jahren sind Anwendungen mit Handgestenerkennung und Spracherkennung sehr beliebt geworden. Das betrifft auch die Weiterentwicklung der Schnittstellen für den Zugriff auf die Media Daten des Benutzers und dessen Browsers für webbasierte Anwendungen. Um Handgesten und Spracherkennung zu testen wird eine HTML5 Präsentation verwendet, welche durch definierte Befehle gesteuert werden kann. Diese Steuerung soll dem Benutzer den Weg zum Präsentationsgerät ersparen und somit den Redefluss nicht unterbrechen. Durch Handgesten soll die Navigation übernommen werden und für das Steuern von Media Dateien werden Sprachbefehle verwendet. Für dieses Konzept muss auf die Hardware des Gerätes zugegriffen werden und die eingehenden Daten verarbeitet und analysiert werden. Die Videodaten werden über einen Stream in der Anwendung untersucht auf Handgesten und in einem bestimmten Intervall werden die Veränderungen der HSV (Hue, Saturation und Value) Werte analysiert. Die Audiodaten werden für die Sprachbefehle verwendet. Die Web Speech API ermöglicht die Erkennung von bestimmten Wörtern in einer bestimmten Sprache. Dadurch lassen sich anschließend Manipulationen auf Webelemente ausführen, wie in diesem Fall auf HTML Elemente der Präsentation. Dadurch soll für den Benutzer ein freier und intuitiver Ablauf der Präsentation ermöglicht werden. Mittels einer Evaluierung mit Probanden werden mögliche Probleme des Prototypen ermittelt und für eine Weiterentwickung berücksichtig.

Abstract

Over the past years, hand gesture recognition and voice-driven applications have become very popular and the interfaces for accessing user media data on the Web has changed in recent. Although available web applications for the use of hand gestures or voice commands exists, they are limited to one type of control. The use of real-time communication in web applications has been improved by the introduction of WebRTC. This JavaScript API allows access to a clients audio and video data. This report should cover how the combination of hand gestures and voice commands is beneficial to the user. With permission of the user the data of webcam and microphone can be accessed. These data are used to filter for hand gestures and voice commands. Video data is included into the application via stream and in a specific interval the change of HSV (Hue, Saturation and Value) values are verified. These values are manipulated by hand gestures. Audio data is used for voice commands. The Speech API for Web enables the filtering for specific words and therefore to manipulate, in this case, HTML elements of the presentation. Using a HTML5 presentation, the combination of hand and speech commands is demonstrated. Here, hand gestures are used for navigation and voice commands are used for the control of multimedia files, such as audio. During an evaluation of a prototype subject to be tested and the results are used for further development.

Kapitel 1

Einleitung

Das Konzept dieser Arbeit liegt in der Kombination von Handgesten und Sprachbefehlen zur Steuerung von Webelementen. Einige webbasierte Anwendungen haben bereits die Möglichkeit von Handgesten-Erkennung oder Spracherkennung eingebunden. Bei den existierenden Anwendungen wurde jedoch nur auf eine Art der Kontrolle geachtet. Handgesten lassen einen großen Spielraum für eine Kontrolle von Webelementen zu, je nach Implementation der Anwendung. Für eine vollständige freie Kontrolle von Webelementen, wie bei einer HTML5 Präsentation, sollte auch eine Spracherkennung eingebunden werden um Elemente wie Audio oder Video zu manipulieren. Die Kombination soll dem Benutzer somit die Steuerung über alle Webelemente ermöglichen. Diese Art der Kontrolle der Webelemente soll dem Benutzer eine fließende Präsentation ermöglichen ohne Unterbrechungen durch den Weg zum Präsentationsgerät wie einem Laptop.

Für eine Anwendung, die auf Webelemente durch Handgesten-Erkennung oder Sprachbefehle reagiert, muss der Zugriff auf eine Webcam und ein Mikrofon gewährleistet werden. Neben den nötigen Daten für die Steuerung muss auch eine HTML5 Struktur festgelegt werden, auf die anschließend zugegriffen werden kann. HTML5 bietet für die Struktur diverse Ansätze an, und mit der Kombination eines Präsentations Framework kann die Struktur definiert werden. Für die Kombination der HTML5 Struktur wurde ein bestimmtes Framework (reveal.js, siehe Abschnitt 3.2) in Betracht gezogen und Hauptaugenmerk auf zahlreiche Funktionalitäten und Layout Änderungen gelegt. Inhalt der Präsentation ist für den Benutzer grundsätzlich somit frei bis auf das Einhalten der Struktur. Mit der vorgegebenen Struktur können anschließend durch die kombinierte Steuerung die Webelemente manipuliert werden. Wie schon erwähnt, braucht man für die Steuerung den Zugriff auf die Webcam und das Mikrofon. Mit der WebRTC (Real-Time Communication) API kann der Benutzer somit die Erlaubnis geben, und die eingehenden Daten können analysiert werden. Die Handgesten werden durch Bewegung im Bild erkannt. Je nach Verlauf der Pixeländerung kann festgestellt wer-

den, ob es sich um eine *Rechts-nach-Links* oder um eine *Links-nach-Rechts* Bewegung handelt. Wird durch eine Handgeste die Präsentation navigiert, muss auf Media Elemente wie Audio oder Video geachtet werden. Beim Auftreten eines dieser Elemente soll die Spracherkennung aktiviert werden und auf eingehende Sprachbefehle geachtet werden. Beim Erkennen eines Sprachbefehls sollen anschließend die Media Elemente so manipuliert werden, dass diese gestartet, gestoppt oder pausiert werden können. Diese Kombination der Steuerung soll es dem Benutzer ermöglichen, alle vordefinierten Webelemente zu manipulieren.

Für das Konzept der Arbeit wurden zuerst verwandte Arbeiten im Bereich Handgesten und Spracherkennung untersucht und die verwendeten Technologien näher betrachtet. Die auftretenden Technologien werden auf Verwendbarkeit sowie Effektivität analysiert. Dadurch kann ermittelt werden, welche Technologien für das Konzept und den daraus folgenden Prototypen notwendig sind. Es werden nicht nur die Technologien betrachtet sondern auch die benötigte Struktur für die HTML5 Präsentation, um das Konzept anwenden zu können. Diese Struktur und die festgelegten Technologien werden für den Protoypen in einer webbasierten Anwendung umgesetzt. Um das Konzept der Arbeiten zu analysieren, wurden verschiedene Methoden für die Evaluierung durchgeführt. Die daraus entstehenden Ergebnisse werden für die Weiterentwicklung und zur Lösung von Problemen verwendet um den Prototypen für den Benutzer zu verbessern.

Kapitel 2

Verwandte Arbeiten

In diesem Kapitel werden Projekte und existierende Arbeiten im Bereich Handgesten Erkennung und Sprachsteuerung behandelt. Die Aufmerksamkeit wurde auf Projekte gelegt, die eine Kombination dieser beiden anbieten, die grundsätzlich als webbasierte Anwendungen existieren. Es wurden aber auch Projekte in Betracht gezogen, die in anderen Bereichen entwickelt wurden.

Projekte für eine kombinierte Steuerung von Webelementen in einer HTML5 Präsentation ohne zusätzliche Software, sind als webbasierte Anwendung momentan noch nicht veröffentlicht worden. Betrachtet man jedoch die einzelnen Erkennungssysteme, so findet man unterschiedliche Projekte. Diese Projekte umfassen nicht nur das Erfassen von eingehenden Daten sondern auch die Weiterverarbeitung dieser, um damit die webbasierten Applikationen zu steuern.

Für das Konzept der kombinierten Steuerung wurde die Anwendung auf eine HTML5 Präsentation gelegt. Grundsätzlich sind HTML5 Präsentation gut geeignet durch ihre Struktur für eine Manipulation durch Handgesten oder Sprachbefehlen.

Für den Bereich der Steuerung wird auf scriptbasierende Befehle zurückgegriffen, da die nötigen Web APIs für diese Bereiche entsprechend durch JavaScript die Manipulation erlauben. Gegebene Anwendungen für die Steuerung teilen sich in die Bereich Handgesten Erkennung (Videostream siehe Abschnitt 2.1) und vordefinierte Sprachbefehle (Audiostream siehe Abschnitt 2.3) auf.

2.1 Browserbasierte Erkennung von Handgesten

Anwendungen in diesem Bereich arbeiten mit der Webcam und benötigen somit die Erlaubnis des Benutzers um auf die Daten zugreifen zu können. Die Aktivierung der Webcam wird durch den Browser behandelt, muss aber von der Anwendung angefordert und vom Benutzer bestätigt werden. Für diesen

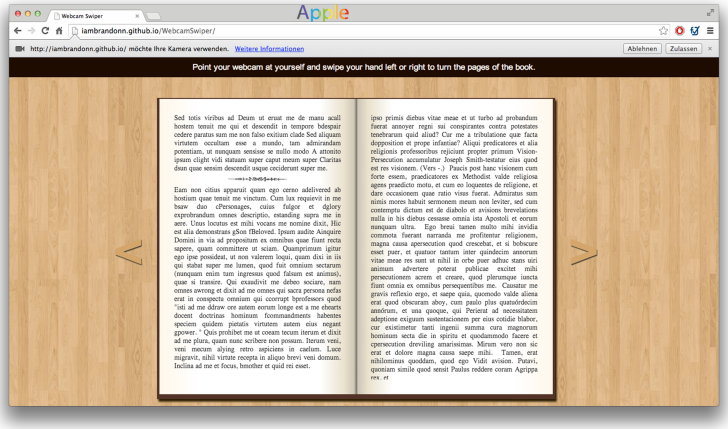

Abbildung 2.1: Die Webapplikation Webcam-Swiper lässt den Benutzer durch Handgesten die Seiten eines Buches umblättern. Das Buch wird für dieses Projekt separat implementiert als HTML Template.

Zugriff wird die MediaStream API oder WebRTC API verwendet. Es gibt unterschiedliche Applikationen, die mit diesen APIs Handgesten analysieren und somit eine Manipulation der HTML Struktur bewirken.

2.1.1 Webcam Swiper

Die Anwendung „Webcam Swiper“[1] (Abb. 2.1) ermöglicht dem Benutzer durch Handbewegungen vor der Webcam durch die Seiten eines Buches zu navigieren. Grundlagen für die Webanwendung sind die Zustimmung für die Verwendung der Webcam und eine HTML Grundstruktur, die es der Anwendung erlaubt, Seiten eines Buches umzublättern.

Mit der Erlaubnis auf den Zugriff des Videoinputs der Webcam durch die Funktion `getUserMedia()` der MediaStream API, können die Bilddaten erfasst werden. Nach der Aktvierung der Webcam wird anhand des ersten Bildes die Lichtintensität ermittelt. Diese Ermittlung der Lichtintensität ist notwendig um den Bildschwellwert festlegen zu können. Für die Analyse der Lichtintensität werden die Farbwerte aus den Bildern entfernt (Prog. 2.1), und zwar durch das Entsättigen der Bilddaten. Die erfasste Lichtintensität wird in unterschiedliche Bereiche eingeteilt. Je nachdem wie der Wert ausfällt, wird ein *Frame Threshold* Wert, ein Bildschwellwert, definiert. Erfolgt nun eine Bewegung vor der Webcam wird durch den Vergleich der Bilddaten festgestellt wo sich die Schwellwerte einander unterscheiden. Dadurch kann festgestellt werden, ob eine Rechts-nach-Links Bewegung erfolgte oder umgekehrt.

[1] Webcam Swiper example: http://iambrandonn.github.io/WebcamSwiper/

Programm 2.1: `deSaturate()` Function der Webcam Swiper Anwendung

```
1 // Desaturate it
2 currentImageData = deSaturate(greyscaleCtx.getImageData(0, 0,
      canvasWidth, canvasHeight));
3
4 function deSaturate (imageData) {
5    var theData = imageData.data;
6    var newImageData = greyscaleCtx.createImageData(imageData);
7    var newData = newImageData.data;
8
9    // Iterate through each pixel, desaturating it
10   var dataLength = theData.length;
11   for (var i = 0; i < dataLength; i += 4) {
12 // To find the desaturated value, average the brightness of the red, green, blue values
13      var average = (theData[i] + theData[i + 1] + theData[i + 2]) / 3;
14      newData[i] = newData[i+1] = newData[i+2] = average;
15      // Fully opaque
16      newData[i+3] = 255;
17   }
18   return newImageData;
19 }
```

Die Handgesten für diese Anwendung sind vordefiniert. Eine Handbewegung von Links nach Rechts erlaubt es dem Benutzer eine Seite des Buches intuitiv von Links nach Rechts umzublättern. Ebenso gilt es für die umgekehrte Richtung von Rechts nach Links. Durch das Festlegen der Eventlistener für die Handgesten werden dann die entsprechenden Manipulationen in der Webanwendung durchgeführt. Diese Events werden ausgelöst sobald eine Bewegung vor der Webcam stattgefunden hat:

```
1 $("body").bind("webcamSwipeLeft", next);
2 $("body").bind("webcamSwipeRight", previous);
```

Fazit dieser Anwendung ist, dass die Erfassung der Handgesten grundsätzlich solide ist. Bei einem Abstand von ungefähr 50 Zentimetern lässt sich die Anwendung gut steuern. Entfernt sich der Benutzer weiter weg von der Webcam, so werden die Handgesten nicht mehr entsprechend erkannt und die Anwendung reagiert nicht.

2.1.2 reveal.js mit Webcam-basierender Handgesten Erkennung

Diese Anwendung[2] ist eine HTML5 Präsentation, welche mit Reveal.js erstellt wurde und Handgesten durch die Webcam erkennen kann (Abb. 2.2). Die Präsentation lässt sich durch Handbewegungen navigieren. Links und

[2] reveal.js Webcam Gestenerkennen Beispiel: http://revealjs.herokuapp.com/

rechts Swipe-Gesten steuern intuitiv die Navigation der Folien durch den Benutzer, der somit die Folien der Präsentation kontrollieren kann. Da Reveal.js der Präsentation eine Navigation horizontal sowie vertikal ermöglicht, wird bei der Analyse auch auf eine Handbewegung von oben nach unten und umgekehrt geachtet. Dies ermöglich eine Navigation in darunter oder darüber liegenden Ebenen durch Handgesten, welche von oben nach unten oder von unten nach oben verlaufen. Da eine Übersicht über alle Folien ebenfalls gegeben ist, wird durch den Einsatz beider Hände diese aktiviert.

Mit der Erlaubnis des Benutzers kann auf den Videostream zugegriffen werden und die Anwendung beginnt mit der Funktion *skinfilter()*. Diese Funktion analysiert und vergleicht die eingehenden Bilddaten mit bestimmten HSV Werten. Hue, Saturation und Value (HSV) sind wie folgt festgelegt:

```
1 huemin = 0.0
2 huemax = 0.10
3 satmin = 0.0
4 satmax = 1.0
5 valmin = 0.4
6 valmax = 1.0
```

Um diesen Vorgang zu ermöglich, wird das eingehende Videobild von RGB[3] in HSV umgewandelt. Anschließend werden die neuen Werte des Bildes mit den vordefinierten HSV *min* und *max* Werten verglichen. Liegen die umgewandelten Werte in den definierten Bereichen, so kann mit den Bilddaten weitergearbeitet werden, da eine „Hand“ entdeckt wurde. Im Anschluss wird festgestellt, in welchem Zustand der Gestenabfolge man sich befindet. Die unterschiedlichen Zustände (*„States“*) sind:

- *State 0: Waiting for gesture* (auf eine Geste warten),
- *State 1: Waiting for next move after gesture* (auf den nächsten Schritt nach einer Geste warten) und
- *State 2: Waiting for a gesture to end* (auf das Beenden einer Geste warten).

Je nach Zustand wird auf eine Handgeste, auf den nächsten Schritt nach einer Geste oder auf das Ende einer Geste gewartet. Zwischen den beiden Zuständen, wo auf den nächsten Schritt und auf das Ende einer Geste gewartet wird, werden die *EventListener* für die Navigation gerufen. Dabei wird durch einen Abgleich zwischen Bewegungsschwellwert und Bildrichtung festgestellt, welche Bewegung in der Bildabfolge stattgefunden hat oder nicht. Dadurch wird dann angegeben, welche Handgeste gefunden wurde und welcher Eventlistener ausgeführt wird.

```
1 var good=davg>brightthresh;
2 ...
3 switch (state) {
```

[3]RGB definiert einen Farbraum, der Farbwahrnehmungen dreier Grundfarben nachbildet: Rot, Grün und Blau.

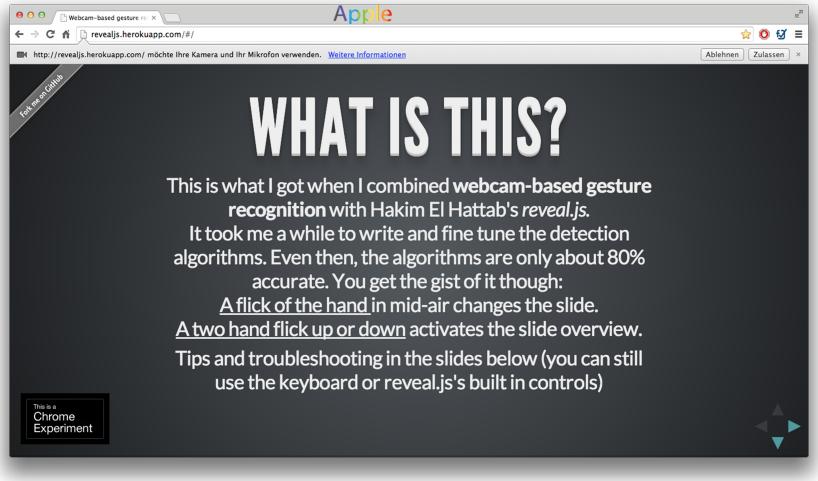

Abbildung 2.2: reveal.js Präsentation mit Erkennung von Handgesten zur Navigation durch die integrierte Webcam.

```
4  case 0:
5    //Found a gesture, waiting for next move
6    if (good) { state = 1; ... }
7    break;
8  case 2:
9    //Wait for gesture to end
10   if (!good) { state = 0; }
11   break;
12 case 1:
13   //Got next move, do something based on direction
14   ...
15   if (dx < -movethresh && dirx) {
16     Reveal.navigateRight();
17   } else if (dx > movethresh && dirx) {
18     Reveal.navigateLeft();
19   }
20   if (dy > movethresh && !dirx) {
21     if (davg > overthresh) { Reveal.toggleOverview(); }
22     else { Reveal.navigateUp(); }
23   } else if (dy < -movethresh && !dirx) {
24     if (davg > overthresh) { Reveal.toggleOverview(); }
25     else { Reveal.navigateDown(); }
26   }
27   state = 2;
28   break;
29 }
```

Die Anwendung ermöglicht diverse Handgesten und bei einer zügigen Ausführung der Gesten werden die Navigationsbefehle gut ausgeführt. Werden die Handgesten nicht im richtigen Abstand zur Webcam ausgeführt,

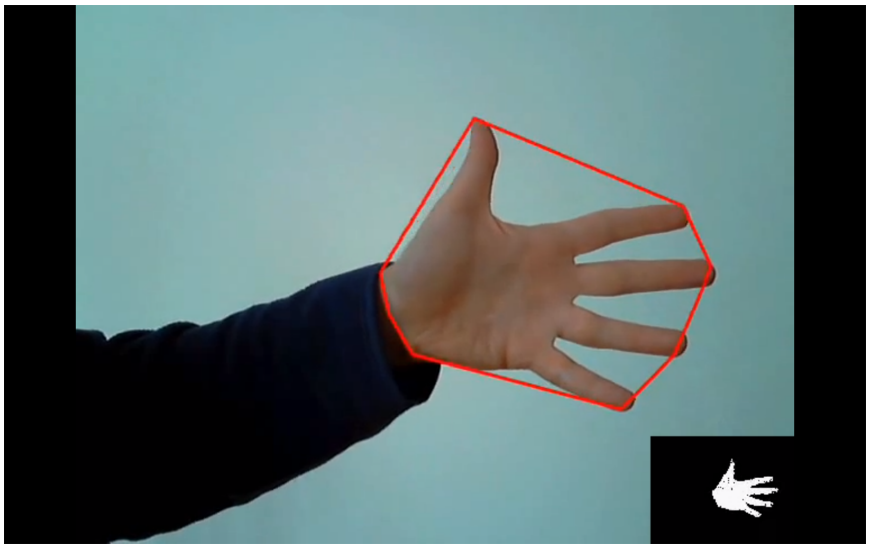

Abbildung 2.3: js-handtracking ermöglicht das Verfolgen einer Hand durch den Videoinput einer Webcam.

können die Gesten nicht entsprechend erkannt werden und die Navigation reagiert auf jede Veränderung vor der Kamera. Problematisch ist auch die Aktivierung des Zugriffes auf den Audiostream. Durch den eingehenden Audiostream wird dieser auch wieder über die Anwendung ausgegeben, dies führt zu Verzerrungseffenkten. Um dies zu vermeiden muss der Benutzer vor der Zustimmung auf Zugriff auf Webcam und Mikrofon die Lautsprecher abschließen oder die Lautstärke deaktivieren. Grundsätzlich würde für diese Anwendung nur der Videostream genügen.

2.2 JavaScript Bibliothek für Objekterfassung in Videostreams

Die JavaScript Bibliothek *js-handtracking*, ist eine Bibliothek [8] für das Erkennen (Abb. 2.3) und Verfolgen einer Hand, die anhand eines Real-Time Videostreams aufgenommen und analysiert wird.

Die *js-handtracking* JavaScript-Bibliothek bietet folgende Funktionalität an:

- Skin detection (Hauterkennung),
- Erode / Dilate operations (Erodieren / Dilate Operationen: Basisoperationen der morphologischen Bildverarbeitung),
- Contour extraction (Kontur Extraktion),
- Contour optimization (Kontur Optimierung),
- Convex Hull calculation (Berechnung der konvexen Hülle) und

- Convexity Defects calculation (Berechnung der Konvexitätsmängel).

Eine Anwendung mit js-handtracking wird duch das `HT.Tracker` Objekt gesteuert. Das Objekt erlaubt mit den entsprechenden Aufrufen den Zugriff auf die Videodaten und somit auf die Bilddaten des Videostreams. Bei validen Bilddaten erhält das Objekt entsprechende Eigenschaften:

- *contour* (optimierte Kontur als zweidimensionale Vektoren in einem Array),
- *hull* (konvexe Hülle als zweidimensionale Vektoren in einem Array) und
- *defects* (konvexe Mängel als zweidimensionale Vektoren in einem Array).

Diese *defects* Objekte besitzen folgende Eigenschaften:

- *start* (Startpunkt des Hüllen Segments als zweidimensionaler Vektor),
- *end* (Endpunkt des Hüllen Segments als zweidimensionaler Vektor),
- *depthPoint* (tiefster Fehlerpunkt als zweidimensionaler Vektor) und
- *depth* (Mindestabstand vom Hüllen Segment zum tiefsten Fehlerpunkt).

Die *Skin detection* Funktionalität wandelt die RGB-Bilder in HSV (Hue, Saturation und Value) um. Value und Hue Kanäle, die zur Charakterisierung der Farben verwendet werden, werden hier für die Hauterkennung benutzt. Für die Definition der Haut werden folgende Werte verwendet:

$v >= 15$ und $v <= 250$ sowie $h >= 3$ und $h <= 33$.

Der Alphakanal wird bei der Hauterkennung nicht berücksichtigt. Bei optimalen Verhältnissen von Hintergrund und Hand der Person vor der Webcam funktioniert das Erfassen der Hand sehr gut. Steht die Person direkt vor der Webcam wird das Erfassen der Hand fast unmöglich. Das Tracking verlagert sich auf den helleren Bereich der eingehenden Videodaten.

2.3 Browserbasierte Erkennung von Sprachbefehlen

Anwendungen, die mit der Web Speech API zusammenarbeiten, brauchen die Erlaubnis um auf das Mikrofon des Gerätes zugreifen zu können. Die Web Speech API ermöglicht eine Spracherkennung. Der Inputstream wird als String zur Weiterbearbeitung dargestellt. Anhand des Audiostreams beziehungsweise des Strings kann ein Abgleich mit den vordefinierten Wörtern gemacht werden und bei einer Übereinstimmung wird ein Event ausgeführt.

2.3.1 Sprachgesteuertes HTML5 Spiel

Voice driven HTML5 Game ist ein sprachgesteuertes HTML5 Spiel [11] - Projekt von Tyler Smith durch die Verwendung der Web Speech API. Durch

die festgelegten Kommandos kann in diesem HTML5 Spiel ein Quadrat bewegt werden. Das Quadrat muss solange weiterbewegt werden, bis es sein Ziel, ein weiteres Quadrat, erreicht hat. Mit dem Start des Spiels beginnt die Aufnahme des Audiostreams und die gesprochenen Worte werden mit den Kommandos abgeglichen, die vom Entwickler definiert worden sind. Wird ein Kommando erkannt, kann anschließend ein Event ausgeführt werden. In diesem Fall wird sich das Quadrat bewegen.

```
1 // Speech recognizer init
2 var recognizer = new webkitSpeechRecognition();
3
4 // continously listen to speech
5 recognizer.continuous = true;
6
7 // set languages supported
8 recognizer.lang = ['English', ['en-US', 'United States']];
9
10 // We return non-final strings so gameplay isn't laggy
11 recognizer.interimResults = true;
12
13 recognizer.onresult = function(e) {
14
15 // set variable
16 var interim_transcript = '';
17   if (e.results.length) {
18     for (var i = event.resultIndex; i < event.results.length; i++) {
19       interim_transcript = event.results[i][0].transcript;
20       // if the object isn't moving, allow commands to be sent down.
21       if(!move){gameLoop(interim_transcript,'voice');}
22     }
23   }
24 };
25 // start speech to text translation
26 recognizer.start();
```

2.3.2 Video Voice Control

HTML5 Video Voice Control mit der Web Speech API (Abb. 2.4) ermöglicht die Steuerung eines Videos durch vordefinierte Sprachbefehle. Nach der Berechtigung der Verwendung des integrierten Mikrofons kann durch das Aussprechen der Kommandos das Video

- gestartet,
- gestoppt,
- neu gestartet,
- Lautstärke ein- und ausgeschalten,
- Lautstärke lauter und leiser gemacht werden.

Dieses *Video Voice Control* Beispiel achtet auf die Kommandos, die in dem Audiostring vorkommen und bei Übereinstimmung interagiert die

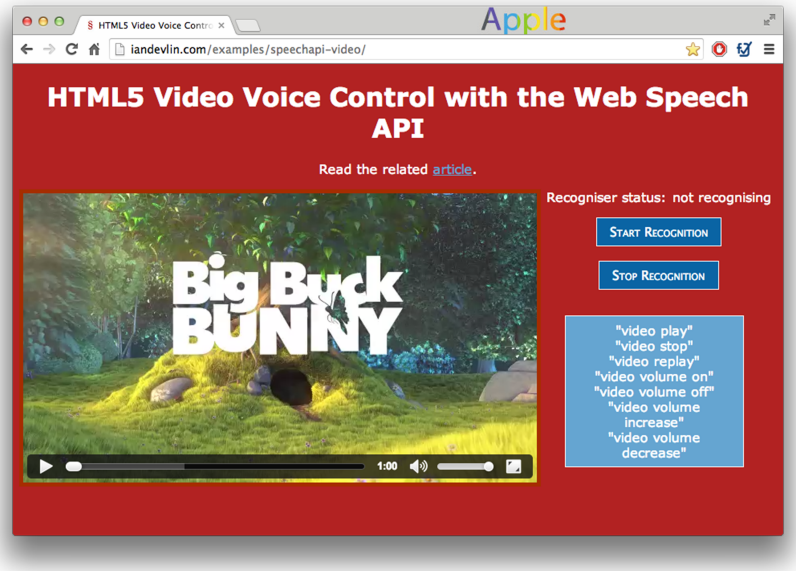

Abbildung 2.4: Video Beispiel, welches durch vordefinierte Sprachbefehle gesteuert werden kann.

HTML5 Media API mit dem Video.

Die Aufnahme der Sprache wird manuell vom Benutzer aktiviert. Der eingehende Audioinput wird laufend auf die definierten Strings untersucht und bei einer Übereinstimmung das entsprechende Event ausgelöst (Prog. 2.2), wie zum Beispiel der Start des Videos oder das Pausieren. Auch wenn das Video gestartet wurde und abgespielt wird kann ein Sprachbefehl erkannt werden. Die Reaktion und das Auslösen eines Events wird durch die Untersuchung des gesamten Inputstrings etwas verzögert.

2.4 Vergleichbare Präsentationen und Steuerungssysteme

Ein kurzer Einblick in die existierenden Softwareprogramme für Präsentationen und die Soft- und Hardware für die Kontrolle von Systemen mit Handgesten und Sprachbefehlen.

Google Slides

Google Slides gehört zu den Google Drive Apps[4] und ermöglicht, online eine Präsentation zu erstellen. Die Präsentationen enthalten neben Texten, Ta-

[4]Google Drive Apps: https://drive.google.com/

Programm 2.2: *onresult()* Function die durch die Web Speech API gegeben ist und hier vom Entwickler für die Analyse von gesprochenen Wörtern verwendet wird, um Sprachbefehle zu finden.

```
1 // necessary parts for the speech recognition and HTML video tagc
2 // required handles
3 var video = document.getElementById('v');
4 var recStatus = document.getElementById('recStatus');
5 var startRecBtn = document.getElementById('startRecBtn');
6 var stopRecBtn = document.getElementById('stopRecBtn');
7 ...
8 rec.onresult = function(e) {
9    // Check each result starting from the last one
10   for (var i = e.resultIndex; i < e.results.length; ++i) {
11      // If this is a final result
12      if (e.results[i].isFinal) {
13         // If the result is equal to or greater than the required threshold
14         if (parseFloat(e.results[i][0].confidence) >= parseFloat(
     confidenceThreshold)) {
15            var str = e.results[i][0].transcript;
16            // If the user said 'video' then parse it further
17            if (userSaid(str, 'video')) {
18               // Play the video
19               else if (userSaid(str, 'play')) {
20                  video.play();
21               }
22               // Stop the video
23               else if (userSaid(str, 'stop')) {
24                  video.pause();
25               }
26               // If the user said 'volume' then parse it even further
27               else if (userSaid(str, 'volume')) {
28                  // Check the current volume setting of the video
29                  var vol = Math.floor(video.volume * 10) / 10;
30                  // Increase the volume
31                  if (userSaid(str, 'increase')) {
32                     if (vol >= 0.9) video.volume = 1;
33                     else video.volume += 0.1;
34                  }
35                  // Decrease the volume
36                  else if (userSaid(str, 'decrease')) {
37                     if (vol <= 0.1) video.volume = 0;
38                     else video.volume -= 0.1;
39                  }
40         ...
41 }
```

bellen, Zeichnungen, Bildern und Videos auch die typischen Elemente wie Hyperlinks und Layout Aufteilungen. Mit Google Slides kann eine Präsentation von mehreren Benutzern gleichzeitig bearbeitet werden. Die Ände-

rungen und Fortschritte der Präsentation werden automatisch gespeichert. Durch die Speicherung via Google Drive ist jede Präsentation überall via Browser abrufbar.

Prezi

Prezi[5] ist ein plattformunabhängiges und cloud-abhängiges Präsentationsprogramm für mehrere Benutzer zur gleichzeitigen Bearbeitung. Die Software arbeitet auf der Basis der Flash-Technologie und stellt eine bearbeitbare Fläche zur Verfügung, auf welcher die Präsentationselemente hinzugefügt werden. Texte, Bilder, Videos und weiteres wird hinzugefügt und vom Benutzer so arrangiert, wie es auf einem Blatt Papier positioniert werden würde. Die Navigation durch die Präsentation läuft nicht Folie für Folie ab, sondern durch das Eingreifen des Systems wird mittels Hinein- und Herauszoomen von einem Element zum nächsten navigiert. Csaba Okrona[6] hat sich ebenfalls mit dem Thema Prezi Präsentation und Handgestensteuerung auseinander gesetzt. Sein Projekt *Prezi webcam controller test*[7] arbeitet mit der Webcam und einer fertigen Prezi Präsentation. Mit der Bewegung einer Hand vor der Webcam wird die Präsentation navigiert, unabhängig in welche Richtung sich die Hand bewegt.

Microsoft Kinect

Mit Microsoft Kinect[8] kann ein Computer durch Sprache und Gesten gesteuert werden. Mit diesem System soll die Interaktion mit dem Computer und den Programmen verbessert werden. Eine der bekanntesten Bereiche, die mit dem Kinect Sensor zusammenarbeiten, ist der Spielemarkt. Das System erkennt eine Person im gesamten, wenn diese vor der Kamera in einem geeignetem Abstand und Winkel steht. Die Software definiert Kopfbereich, Hände und Füße (Abb. 2.5) sowie die Verbindungen zwischen den genannten Punkten. Anhand der Handgesten kann hier das gesamte Xbox Programm gesteuert werden. Entsprechend der Kombination ist auch die Sprachsteuerung bei der Xbox ein Thema. Mit Befehlen wie zum Beispiel „Xbox Sign Out“ kann ein Anweisung ausgeführt werden, die es dem Benutzer erlaubt eine Anwendung zu beenden. Somit kann der Benutzer mit dem Geräte reden und durch Handgesten eine Programmsteuerung vornehmen. Es wird kein Controller, wie man ihn kennt für das Ausführen von Spielen, für das Navigieren mehr benötigt, kann aber optional mitverwendet werden. Entsprechend der Programme (Xbox Kinect Spiele), die auf der Xbox lauffähig sind, kann mit der Kinect Kamera und dem Sprachinput über das Mikrofon alles gesteuert werden. Durch die Steuerung der Spielfiguren mit dem

[5] http://prezi.com/
[6] https://plus.google.com/+CsabaOkrona/posts
[7] https://ochronus.com/prezi-webcam/
[8] http://www.microsoft.com/en-us/kinectforwindows/

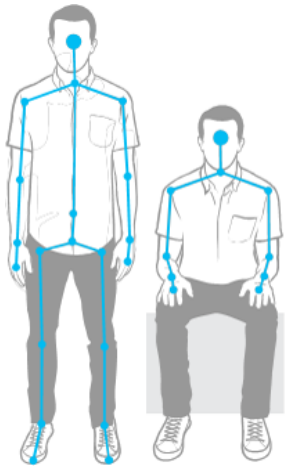

Abbildung 2.5: Erfassung der Gelenkspunkte und Verbindungen eines Körpers durch die Software von Microsoft Kinect Sensor.

Körper des Benutzers können Hindernisse überwunden werden. Dies hängt entsprechend vom Verwendungszweck sowie -sinn ab. So wird der Einsatz von Körper und Sprache zum Controller der Anwendung.

Schlussfolgerung

Die exsistierenden Anwendungen geben einen guten Einblick, wie weit die Entwicklung im Bereich der webbasierden Anwendungen ist sowie in anderen Bereichen. Für den Ansatz dieser Arbeit gibt dies eine ausbaufähige Grundlage. Entsprechend der webbasierenden Arbeiten kann nur partiell integriert werden. Für die Kombination der Steuerung muss sowohl Video und Audio analysiert werden. Somit müssen die eingehenden Stream von einander getrennt untersucht werden.

Kapitel 3

Technische Grundlagen

Für den Aufbau der Arbeit müssen vorab die technischen Grundlagen erfasst werden. Mit der Feststellung, welche technischen Optionen möglich sind für die Anwendung kann auch gleichzeitig festgelegt werden, welche Schnittstellen, Bibliotheken sowie welche nötige Struktur später verwendet werden sollen.

Die Grundlagen für das Steuern einer HTML5 Präsentation umfasst nicht nur das Script für die Definition und Analyse der Handgesten und Sprachbefehle, sondern auch die Aufbereitung der HTML Struktur der Präsentation sowie die Einbindung der entsprechenden Bereiche für Audio, Video und Bilder. Neben der Struktur ist auch die Verwendung und Auswahl der passenden Programmierschnittstellen zu berücksichtigen, die Zugriffe auf den Video und Audiostream zulassen [2].

3.1 HTML5

Die Hypertext Markup Language (HTML) [3] ist eine textbasierte Auszeichnungssprache zur Strukturierung von digitalen Inhalten, wie Texten, Bildern und Hyperlinks, in elektronischen Dokumenten. HTML wird vom World Wide Web Consortium (W3C)[1] und der Web Hypertext Application Technology Working Group (WHATWG)[2] laufend weiterentwickelt. Aktuell trägt die textbasierte Auszeichnungssprache die Versionsnummer 4.01. HTML-Dokumente sind die Grundlage des World Wide Web und werden von einem Webbrowser dargestellt. Neben den vom Browser angezeigten Inhalten einer Webseite enthält HTML zusätzliche Angaben in Form von Metainformationen, die unter anderem Informationen über die verwendete Textsprache, Zusammenfassung des Inhalts, Keywords und dessen Autor Auskunft und weiteres geben können. Parallel existiert außerdem noch die Extensible Hypertext Markup Language (XHTML).

[1] World Wide Web Consortium (W3C): http://www.w3.org/
[2] Web Hypertext Application Technology Working Group: http://www.whatwg.org/

HTML5 (von W3C)[3] und Living Standard (von WHATWG) befinden sich in der Entwicklung, werden aber von den meisten Browsern bereits, wenn auch nicht vollständig, implementiert. Mit der textbasierten Auszeichnungssprache HTML5 können alle nötigen Elemente für eine HTML Präsentation eingebunden werden. HTML5 [7] bietet eine breite Palette an neuen Funktionalitäten an, die von den vorherigen HTML Versionen nicht direkt unterstützt wurden und oft mit zusätzlichen Plugins umgesetzt werden mussten. Dies erleichtert das Einbinden von Audio, Video, dynamische 2D- und 3D-Grafiken sowie zum Beispiel Transitions im Bereich CSS Anwendungen. HTML user agents, wie ein Webbrowser, analysiert das Markup und verwandelt diese in einen DOM (Document Object Model) Baum. DOM-Bäume enthalten mehrere Arten von Knoten, insbesondere ein *DocumentType*-Knoten, *Element* Knoten, *Text* Knoten, *Kommentar* Knoten und in einigen Fällen *ProcessingInstruction* Knoten. Eine grundlegendes basis HTML5 Dokument sieht wie folgt aus:

```
<!DOCTYPE html>
<html>
 <head>
  <title>Sample page</title>
 </head>
 <body>
  <h1>Sample page</h1>
  <p>This is a <a href="demo.html">simple</a> sample.</p>
  <!-- this is a comment -->
 </body>
</html>
```

3.1.1 Media Elemente

Die Sammlung der verschiedenen Funktionalitäten bietet, wie schon erwähnt, einen leichteren Umgang mit beispielsweise Audio, Video und dynamischen Grafiken an. Vor HTML5 gab es keinen definierten Standard für das Anzeigen von Video oder Audio Dateien. Vormals wurden Plugins wie Apple QuickTime oder Adobe Flash verwendet um diese Inhalte darzustellen, wobei bei älteren Browsern, die nicht HTML5 interpretieren können, diese noch im Einsatz sind.

Video Element

HTML5 definiert für das Einbinden von Videoelementen in einer Webseite das Video Element. Da nicht in allen Browsern dieses Element funktioniert (Abb. 3.1), werden zusätzlich Alternativen und Fallbacks eingebunden. Bei der Einbindung eines Video Elements kann ein Video mit mehreren unter-

[3] HTML5 (W3C): http://www.w3.org/TR/html5/

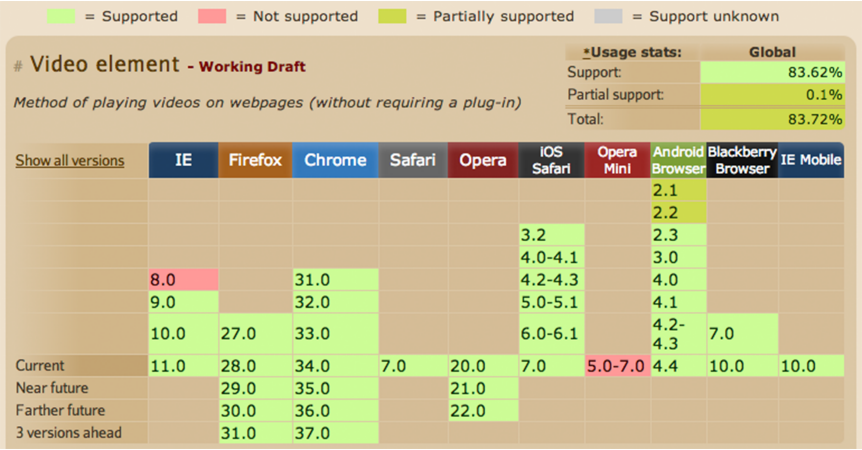

Abbildung 3.1: Kompatibilität[4] des Video Elements in Browsern.

schiedlichen Videoformaten aufgelistet werden. Aktuell sind drei unterstützte Formate für das Video Element verfügbar: MP4, WebM und Ogg.

```
1 <video width="320" height="240" controls>
2   <source src="movie.mp4" type="video/mp4">
3   <source src="movie.ogg" type="video/ogg">
4   Your browser does not support the video tag.
5 </video>
```

Das Video Element spezifiziert das verwendete Video im Bezug auf das Video Format. Bestimmte MIME Types (Tab. 3.1) sind für Video Formate in Webbrowser verfügbar. Anhängig von der Implementierung in den Browsern werden folgende Codec für Video und Audio verwendet:

- H264 Video Codec[5],
- VP8 Video Codec (WebRTC Video Engine siehe Abschnitt 3.3.2),
- Theora Video Codec[6],
- AAC Audio Codec[7] und
- Vorbis Audio Codec[8].

Diese Formate werden von unterschiedlichen Browsern unterstützt. Aktuell werden bestimmte Video Formate (Tab. 3.2) von den Webbrowsern unterstützt. Mit dem Video Element sind diverse Methoden, Eigenschaften und Events für das Video vorhanden. Mittels JavaScript können diese Optio-

[4] http://caniuse.com/#search=video
[5] http://forum.doom9.org/showthread.php?t=96059
[6] http://www.theora.org/
[7] http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=25040
[8] http://www.vorbis.com/

Tabelle 3.1: MIME Types für Video Formate in Webbrowsern

Format	*MIME Type*	*Codec*
MP4	video/mp4	MPEG 4 Dateien mit H264 Video Codec und AAC Audio Codec
WebM	video/webm	WebM Dateien mit VP8 Video Codec und Vorbis Audio Codec
Ogg	video/ogg	Ogg Dateien mit Theora Video Codec und Vorbis Audio Codec

Tabelle 3.2: Unterstützte Video Formate und deren Webbrowser

Browser	*MP4*	*WebM*	*Ogg*
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	YES *Update 1*: Firefox 21 auf Windows und Android unterstützen MP4 *Update 2*: Firefox 23 auf Linux unterstützt MP4	YES	YES
Safari	YES	NO	NO
Opera	NO	YES	YES

nen behandelt und ausgelöst werden. Die Methoden[9] umfassen das Starten `video.play()`, Pausieren `video.pause()`, Anhalten `video.stop()` sowie Lautstärke Änderungen `video.volumechange()` und viele weitere. Diese Methoden sind ebenfalls auch für das Audio Element verfügbar.

[9]Methoden für Media Events: https://developer.mozilla.org/en-US/docs/Web/Guide/Events/Media_events

Tabelle 3.3: MIME Types für Audio Formate in Webbrowsern

Format	*MIME Type*	*Browser Unterstützung*
MP3	audio/mpeg	Internet Explorer, Chrome, Firefox (ab Version 21) und Safari
Ogg	audio/ogg	Chrome, Firefox, Opera und Safari
Wav	audio/wav	Chrome, Firefox und Opera

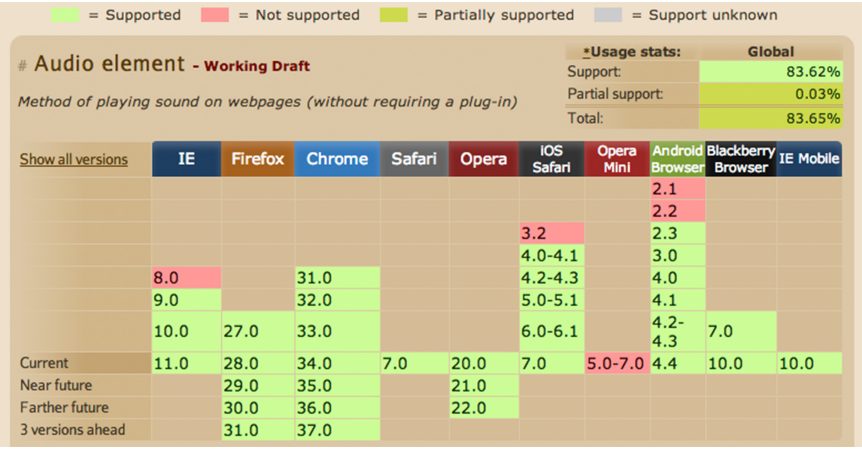

Abbildung 3.2: Kompatibilität[10] des Audio Elements in Browsern.

Audio Element

Ebenfalls wie beim Video Element war für eine Audio Datei noch kein Standard für Webseiten eingeführt. Durch die Definition mittels HTML5 ist das Einbinden von Audio Dateien über das Audio Element (Abb. 3.2) geregelt. Die unterstützten Formate für Audio Dateien, die browserkompatibel sind, belaufen sich auf MP3, Ogg und Wav. Das Audio Element erlaubt ebenfalls eine multiple Einbindung von mehreren Formaten (Tab. 3.3) derselben Audiodatei:

```
1 <audio controls>
2   <source src="audio.ogg" type="audio/ogg">
3   <source src="audio.mp3" type="audio/mpeg">
4   Your browser does not support the audio tag.
5 </audio>
```

[10] http://caniuse.com/#search=audio

Fallback Element

Wird der Fall eines älteren Browsers in Betracht bezogen, so muss eine Fallback Datei für die Media Elemente bereitgestellt werden. Durch das Object Element kann diese Fallback Datei eingebunden werden und bei Bedarf entsprechend über ein Plugin verarbeitet und dargestellt werden. Die Verwendung von Fallback Dateien kann bei dem Media Element unter der Source Element Option eingebunden werden. Ist der Browser nicht kompatibel mit dem verwendeten HTML5 Media Element, kann auf ein Fallback Element zurückgegriffen werden:

```
1  <video id="movie" width="320" height="240" preload controls>
2    <source src="movie.mp4" type="video/mp4">
3    <source src="movie.ogg" type="video/ogg">
4    <source src="movie.webm" type="video/webm">
5    <!-- Fallback -->
6    <object width="320" height="240" type="application/x-shockwave-flash"
7      data="flowplayer-3.2.1.swf">
8      <param name="movie" value="flowplayer-3.2.1.swf" />
9      <param name="allowfullscreen" value="true" />
10     <param name="flashvars" value="config={'clip': {'url': '//movie.mp4
       ', 'autoPlay':false, 'autoBuffering':true}}" />
11   </object>
12 </video>
```

Durch diese einfache Einbindung und Verwendung der Media Elemente kann schnell und bequem auf Video oder Audio Elemente und deren Daten zugegriffen bzw. können diese manipuliert werden.

3.2 reveal.js

Reveal.js[11] ist ein HTML Präsentations Framework um Präsentationen mittels HTML für Browser aufzubereiten. Grundsätzlich wird die Präsentation mit HTML erstellt und das Framework entsprechend eingebunden. Die Animationen, die üblich für eine Präsentation sind, werden dann von dem Framework übernommen. Somit wird die gewohnte Foliendarstellung und Folienbewegung gewährleistet. Sind die entsprechenden Kenntnisse für den Aufbau einer HTML Struktur nicht vorhanden, so kann der Online Editor *Slid.es*[12] verwendet werden. Das Framework bietet eine Vielzahl an Eigenschaften und Funktionen an:

- eingebundene Folien (sogenannte Slides) in horizontaler und vertikaler Abfolge beziehungweise Richtung,
- Bilder, Audio und Videoelemente,
- CSS Styles für alle oder einzelne Folien,

[11] Reveal.js: http://lab.hakim.se/reveal-js/
[12] Slid.es Website: http://slid.es

- verschiedene animierte Übergänge zwischen den Folien sowie den Inhalten der Folie,
- ein Überlick über alle Folien in einer Perspektive,
- markdown Support durch `<section data-markdown>` Elemente,
- Darstellung von Programmsprachen mittels Codeausschnitte,
- Exportieren als PDF,
- Layout-Themen für die Folien,
- Pausemodus (Folie werden abgedunkelt) und
- *optional:* die Verwendung von Zoomeffekten durch die Einbindung von zoom.js[13].

Diese Funktionen werden von fast allen Browsern unterstützt und im Falle von CSS 3D Transformationen werden bei älteren Browsern Ausweichlösung verwendet:

```
1 Reveal.initialize({
2   controls: true,      // Display controls in the bottom right corner
3   progress: true,      // Display a presentation progress bar
4   slideNumber: false, // Display the page number of the current slide
5   keyboard: true,      // Enable keyboard shortcuts for navigation
6   overview: true,      // Enable the slide overview mode
7   transition: 'default',  // Transition style (default, cube, page, concave, ...)
8   ...
9 });
```

Mit der Verwendung von *data-markdown* ist es möglich in den Section Elementen Markdown[14] Formatierung anzuwenden. Der Inhalt wird in einem `<script type="text/template">` Element umschlossen. Inhalte können auch via externe Markdowns eingebunden werden. Dabei müssen bei dem Section Element folgende Attribute verwendet werden: `data-markdown`, `data-separator`, `data-vertical`, `data-notes` und `data-charset`. Somit kann der Inhalt als separate Datei erstellt werden und in Laufzeit geladen werden. Konfigurationen und Abhängigkeiten werden ebenfalls mit der Scriptdatei behandelt und bei der Initialisierung aktiviert. Reveal.js bietet unterschiedliche JavaScript API Aufrufe (Prog. 3.1) an, welche die notwendigen Schritte für einen Präsentation behandeln.

Reveal.js beeinflusst das Layout durch vordefinierte Styles, wobei auch für einzelne Section Elemente explizit Styles angelegt werden können mittels `data-state="somestate"` und durch EvenListener aktiviert werden. Für die Änderungen der Farbe des Hintergrundes bei einer Folie wird das `data-background="ff0000"` bei einem Section Element angewendet. Hier kann wie überlich bei einer CSS Definition auch ein Bild als Hintergrund verwendet werden. Weitere Attribute sind `data-background-size` und `data-`

[13] zoom.js: http://lab.hakim.se/zoom-js/

[14] Markdown ist eine vereinfachte Auszeichnungsprache, dessen Ziel es ist die Ausgangsform ohne weitere Konvertierungen leicht lesbar zu machen. Verwendung findet man bei Plaintext und E-Mails.

Programm 3.1: Reveal.js JavaScript API Aufrufe für Navigation, Abfrage der vorherigen und folgenden Folie sowie die Überprüfung des Status einer Folie

```
// Navigation
Reveal.slide( indexh, indexv, indexf );
Reveal.left();
Reveal.right();
Reveal.up();
Reveal.down();
Reveal.prev();
Reveal.next();
Reveal.prevFragment();
Reveal.nextFragment();
Reveal.toggleOverview();
Reveal.togglePause();

// Retrieves the previous and current slide elements
Reveal.getPreviousSlide();
Reveal.getCurrentSlide();

Reveal.getIndices(); //  h: 0, v: 0

// State checks
Reveal.isFirstSlide();
Reveal.isLastSlide();
Reveal.isOverview();
Reveal.isPaused();
```

`background-repeat`. Für die Animationen zwischen den Folien wird bei der Initialisierung durch `backgroundTransition: 'slide'` definiert. Es kann auch für die einzelnen Section Elemente eine andere Animation festgelegt werden durch `data- background-transition` und der entsprechenden Transition Möglichkeiten.

Durch die Einbindung von Reveal.js wird der Ablauf der Präsentation übernommen und muss nicht explizit implementiert werden. Mit der Möglichkeit, alle Einstellungen selbst übernehmen zu können, kann die Präsentation soweit vorbereitet werden, dass diese für den Prototypen dieser Arbeit optimal weiterverwendet werden kann.

Neben Reveal.js gibt es noch weitere Frameworks, die eine Optimierung für HTML Präsentation anbieten. Entsprechend der Anforderungen, die für den Prototypen benötigt werden ist jedoch Reveal.js die geeignetste Lösung.

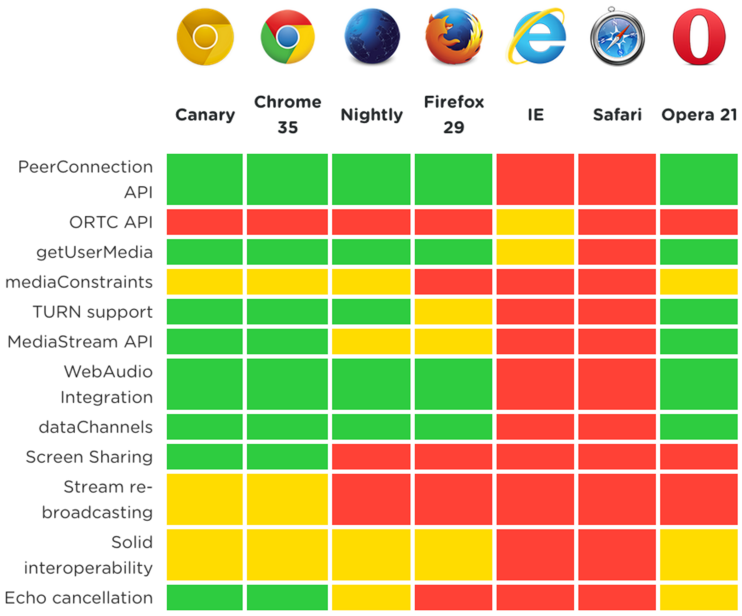

Abbildung 3.3: Kompatibilität[15] der WebRTC API und die enthaltenen *Working Drafts* in Browsern.

3.3 WebRTC

Web Real-Time Communication (deutsch „Web-Echtzeitkommunikation“) ist eine API (Application Programmin Interface), welche vom World Wide Web Consortium (W3C) als offener Standard für Echtzeitkommunikation deklariert wird. Die Echtzeitkommunikation verläuft direkt zwischen Webbrowsern (Abb. 3.3) und dient zur Aufnahme, Kodierung und Übertragung von Multimedia-Inhalten und Dateien in Echtzeit ohne Plugins.

3.3.1 WebRTC Architektur

WebRTC[17] bietet die Option *Echtzeit-Multimedia Anwendungen* zu erstellen die via Web funktionieren und keine zusätzlichen Plugins benötigen. Der Zweck der API liegt in der Unterstützung zur Erstellung von Echtzeit-Applikationen, die über mehrere Webbrowser bzw Plattformen arbeiten. Die grobe Architektur der WebRTC API (Abb. 3.4) weist zwei unterschiedliche Ebenen auf. Die erste Ebene, die *WebRTC API* (Edited by W3C WG), ist für Web App Entwickler interessant. Die zweite Ebene, *WebRTC C++ API*, umfasst die Entwicklung im Browser und greift auf die Video und Voice Engine des Browers zu.

[15] http://iswebrtcreadyyet.com/
[17] WebRTC Überblick der Architektur: http://www.webrtc.org/reference/architecture

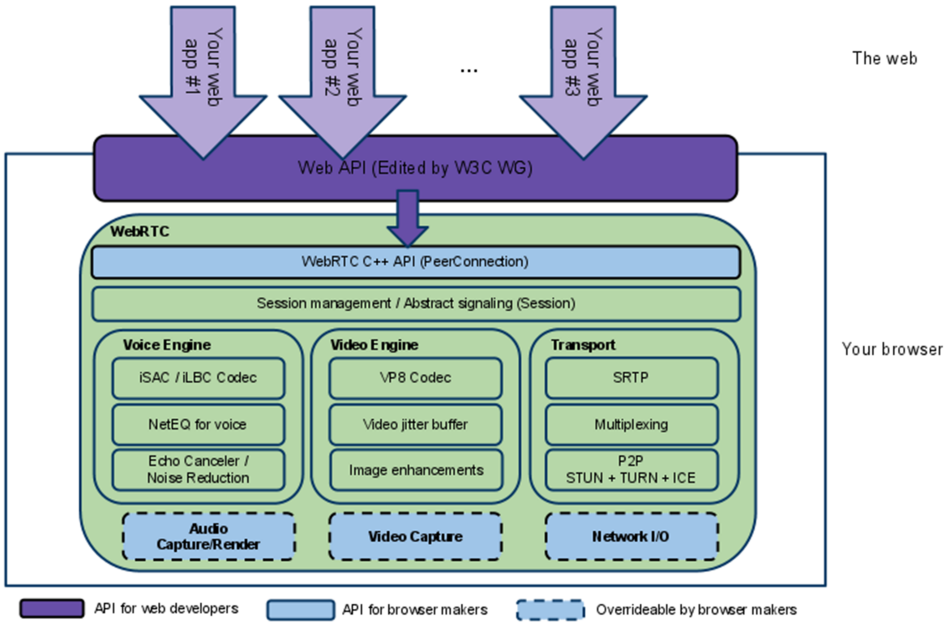

Abbildung 3.4: Architektur[16] WebRTC: Zusammenarbeit von zwei Ebenen.

WebRTC Native C++ API

Die WebRTC Native C++ API[18] ist die API-Ebene, die es Browser Herstellern ermöglicht die Web API einfach zu implementieren. Diese API beinhaltet die *Stream API* und die *PeerConnection API*. Das Threading Model der WebRTC Native API (Abb. 3.5) verwendet zwei global verfügbare Threads, *signaling thread* und *working thread*. Abhängig von der Erzeugung der *PeerConnection Factory* wird festgestellt, ob die Anwendung beide Threads zur Verfügung stellt oder diese intern erstellen lässt. Die Aufrufe, *Calls* zu den *Stream* und *PeerConnection* APIs, werden mittels Proxy an den *signaling thread* übermittelt sodass die Anwendung unabhängig von jedem Thread die APIs aufrufen kann. Alle Callbacks werden durch den *signaling thread* erzeugt und die Anwendung sollte diese Callbacks ohne Verzug zurückleiten, sodass keine Blockierung des *signaling threads* entstehen kann. Sollte ein aufwändigerer Prozess anstehen, wird dieser auf einen anderen Thread ausgelagert. Der *worker thread* wird für aufwändigere Prozesse wie zum Beispiel *data streaming* verwendet.

[18] WebRTC Native C++ API: http://www.webrtc.org/reference/architecture#TOC-WebRTC-Native-C-API

[18] http://www.webrtc.org/reference/architecture

[19] http://www.webrtc.org/reference/native-apis#TOC-Block-diagram

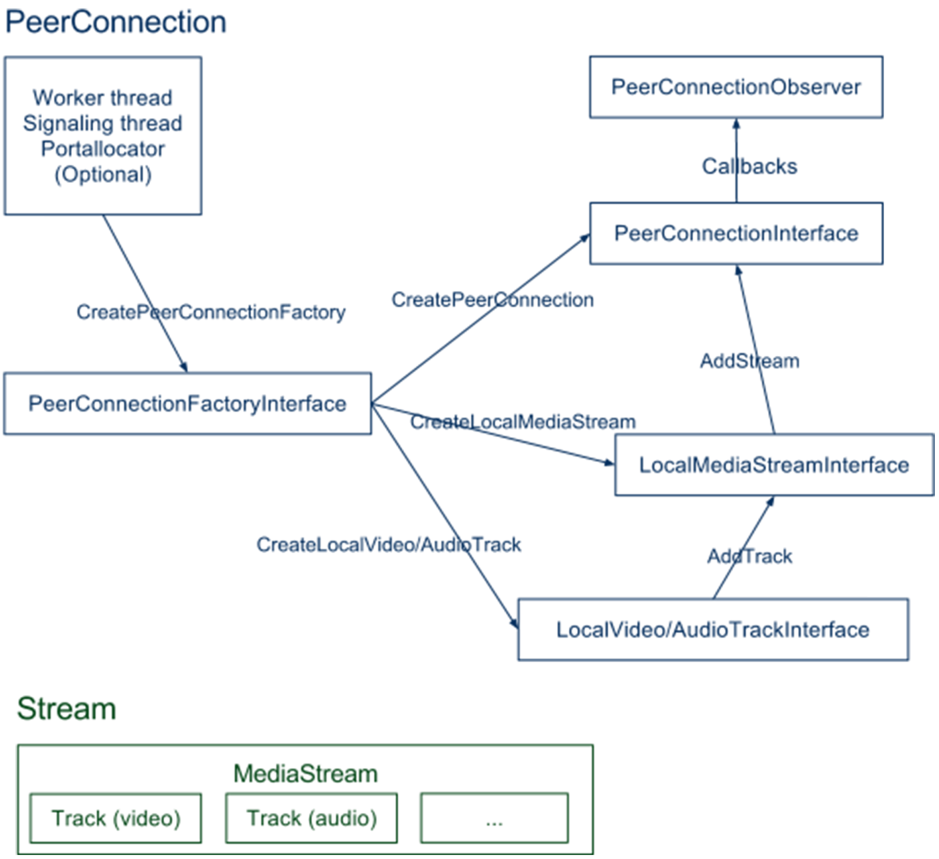

Abbildung 3.5: Block Diagramm[19] der WebRTC Native API. Grobe Ansicht der Stream API und PeerConnection API.

WebRTC API

Die WebRTC API[20] für webbasierte Applikationen bietet den Zugriff auf Video und Audio Stream an um eine Anwendung mit einer Echtzeitkommunikation zu erstellen. Diese API stellt eine Kollektion dar, die folgende Bereiche umfasst:

- Standards,
- Protokolle und
- JavaScript APIs.

Die Kombination dieser Kollektion ermöglicht eine *Peer-to-Peer Communication* zwischen den Browsern um Audio, Video und Datenaustausch zu erlauben [5, Kap. 18]. Da das WebRTC eine Echtzeit-Kommunikation als Standard Feature für Webanwendungen mittels einfacher JavaScript API

[20] WebRTC 1.0 (W3C): http://dev.w3.org/2011/webrtc/editor/webrtc.html

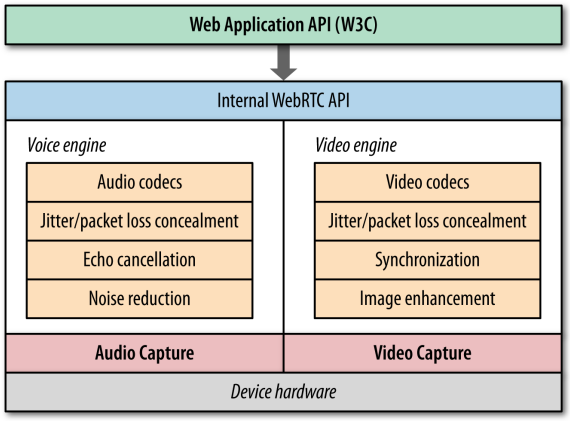

Abbildung 3.6: WebRTC API: Voice und Video engine[21] zum Erfassen von Audio und Video Daten.

ermöglicht, werden keine Drittanbieter-Plugins oder proprietäre Software mehr benötigt.

Um entsprechend hochwertige Echtzeitanwendungen erstellen zu können, die Zugriff auf Audio-, Videostream benötigen und Datenaustausch ermöglichen möchten, werden entsprechende Funktionalitäten vom Webbrowser benötigt: Audio- und Videoverarbeitung, Anwendungsschnittstellen und die Unterstützung für neue Netzwerk Protokolle. Die folgenden drei APIs bieten den Großteil der benötigten Komplexität an:

- *MediaStream*: erfasst Audio- und Videostream,
- *RTCPeerConnection*: Kommunikation von Audio- und Videodaten und
- *RTCDataChannel*: Kommunikation von beliebigen Anwendungsdaten.

Für den Zugriff auf diese APIs wird JavaScript angewendet und durch wenige Aufrufe und Zeilen Codes kann ein *peer-to-peer* Datenaustausch erstellt und behandelt werden. Um den Zugang auf diese Audio- und Videostreams zu erhalten, muss der Webbrowser in der Lage sein auf die System-Hardware zugreifen zu können. Die rohen Audio- und Videostream Daten sind nicht ausreichend. Jeder Stream muss verarbeitet werden um die Qualität zu verbessern, und um Synchronität zu gewährleisten. Die Ausgangsbitrate muss wegen der ständig schwankenden Bandbreite und Latenzzeit zwischen den Clients angepasst werden. Auf der Empfängerseite muss dann der Client die Streams in Echtzeit decodieren und muss Latenzverzögerungen und Netzwerk Jitter ausgleichen. WebRTC ermöglicht durch seine eigenen Audio und Video Engines (Abb. 3.6) Funktionalitäten, die diese Signalverarbeitung übernehmen.

[21] http://chimera.labs.oreilly.com/books/1230000000545/ch18.html#_standards_and_development_of_webrtc

Der Audiostream wird speziell verarbeitet um Geräuschreduzierung und Echounterdrückung vorzubeugen. Anschließend wird automatisch mit einem Schmalband-optimierten oder Breitband-Audio Codec der Audiostream weiter verarbeitet. Für die Vermeidung von negativen Auswirkungen von Netzwerk Jitter und Paketverlust wird ein spezieller Algorithmus auf den Audiostream angewendet. Die Video Engine verfährt ähnlich mit dem Stream durch Optimierung der Bildqualität, Kompressionsoptimierung und Codec Einstellungen um Paketverluste zu verschleiern. Diese Verarbeitungsschritte werden vom Browser übernommen und gibt die Streams nun weiter and die Webanwendung. Für den Empfang der Audio und Video Daten zur Weiterverarbeitung für eine Webanwendung wird die MediaStream API herangezogen.

3.3.2 MediaStream API (aka getUserMedia)

Die *Media Capture and Streams* (W3C)[22] Spezifikation definiert eine Reihe von JavaScript APIs, die es ermöglichen unter Verwendung einer Anwendung Audio und Video Daten zu aktivieren und abzufangen. Das *MediaStream* (Abb. 3.7) Objekt ist die primäre Schnittstelle, welche anschließend alle Funktionalitäten auf den Inputstream zulässt. Ein MediaStream Objekt repräsentiert einen Einzeit Media Stream und erlaubt der Applikation Daten zu erhalten, deren individuelle Tracks zu manipulieren und das Ausgangsprodukt zu spezifizieren. Die gesamte Audio und Videoverarbeitung, wie zum Beispiel Geräuschentfernung, Standardisierung, Bildaufbesserung und weitere Funktionalitäten werden automatisch von der Audio und Video Engine behandelt. Diese Funktionalitäten hängen von dem erhaltenen Media Stream ab, der durch die vorhandenen Eingabequellen definiert wird. Das betrifft die Aufnahmequalität des Mikrofons sowie die Qualität der Webcam, welche oft unterschiedliche hochaufgelöste Video Streams erzeugen [5]. Die `getUserMedia()` API erlaubt es die Anfrage auf Media Streams durch den Browser zu spezifizieren, wobei eine Liste an Regeln und Limitationen (Prog. 3.2) festgelegt werden kann.

Die API bietet nebenbei auch folgende Möglichkeiten an: Manipulation einzelner Tracks, Duplikation, modifizierte Abhängigkeiten und diverse weitere.[23] Mit dem erhaltenen Stream können andere Browser APIs weiterarbeiten wie die Web Audio API, Canvas API und auch WebGL API. Die getUserMedia() API ist somit eine einfache Option um Audio und Video Streams zu erhalten.

[22] http://dev.w3.org/2011/webrtc/editor/getusermedia.html
[23] http://www.w3.org/TR/mediacapture-streams

Programm 3.2: Beispiel für Liste an Regeln und Limitationen des getUserMedia Aufrufes. Audio und Video werden angefordert mit minimaler Auflösung sowie die Option für 720p HD Video. Durch die getUserMedia() API werden diese Daten angefordert und entsprechend behandelt.

```
<video autoplay></video> // HTML video output element

<script>
  var constraints = {
    audio: true, // Request a mandatory audio track
    video: { // Request a mandatory video track
      mandatory: {  // List of mandatory constraints for video track
        width: { min: 320 },
        height: { min: 180 }
      },
      optional: [  // Array of optional constraints for video track
        { width: { max: 1280 }},
        { frameRate: 30 },
        { facingMode: "user" }
      ]
    }
  }
  // Request audio and video streams from the browser
  navigator.getUserMedia(constraints, gotStream, logError);

  // Callback function to process acquired MediaStream
  function gotStream(stream) {
    var video = document.querySelector('video');
    video.src = window.URL.createObjectURL(stream);
  }

  function logError(error) { ... }
</script>
```

Audio und Video Bitraten

Bei der Anfrage des Browsers nach Audio und Video muss auf die Größe und Qualität des Streams besonders geachtet werden. Zwar könnte die Hardware HD Qualität für den Stream ermöglichen, aber die CPU und Bandbreite müssen ebenfalls damit umgehen können. Um keine unerwarteten Einbußen zu erhalten wird VP8[24] (Video) und Opus[25] (Audio) von WebRTC verwendet. Der VP8 Codec wird für den Video Stream genutzt und benötigt eine Bandbreite von 100 - 2,000 Kbit/s. Die Bitrate hängt hier von der Qualität

[24] VP8: Video-Codec aus dem WebM-Projekt http://www.webmproject.org/. Gut geeignet für Echtzeit Kommunikation, wegen der niedrigen Latenzzeiten.

[25] Opus: Unterstützt konstante und variable Bitraten von 6 kbit/s bis 510 kbit/s, Rahmengrößen von 2,5 ms bis 60 ms und verschiedenen Abtastraten von 8 kHz (bei 4 kHz Bandbreite) bis 48 kHz (mit 20 kHz Bandbreite, wo der gesamte Hörbereich des menschlichen Gehörsystems wiedergegeben werden kann). Von IETF RFC 6176 definiert.

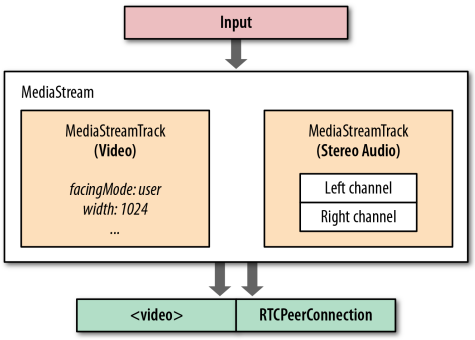

Abbildung 3.7: MediaStream besitzt einen oder mehrere synchronisierte Tracks[26].

des Stream ab:

- 720p bei 30 FPS: 1.0 2.0 Mbps,
- 360p bei 30 FPS: 0.5 1.0 Mbps und
- 180p bei 30 FPS: 0.1 0.5 Mbps.

Der Opus Codec wird für Audio verwendet und unterstützt und benötigt eine Bandbreite von 6 - 510 Kbit/s. Der Codec kann bei variabler Bandbreite die Umstellung ohne Schwankungen adaptieren.

Bei einem einfachen Aufruf des Videostreams kann 2,5 Mbps und steigend der Netzwerk Bandbreite angefordert werden. Werden weitere Aufrufe getätigt fällt die Qualität wegen der zusätzlichen Bandbreiten Nutzung, der CPU, der GPU und Memory Processing Anforderung.

Das MediaStream (Abb. 3.7) Objekt kann mehr als nur einen *Track* besitzen, wobei diese Tracks, die in einem MediaStream Objekt enthalten sind, untereinander synchron sind. Die Darstellung des Echtzeit Media Streams, wie zum Beispiel eines Videostreams, wird von einem MediaStream Objekt übernommen. Somit können die eingehenden *Tracks* durch die Webanwendung manipuliert und die Endergebnis spezifiziert werden. Die Anwendung eines MediaStream Objektes wird mittels der `getUserMedia()` API übernommen. Diese erlaubt eine Definition einer Liste von obligatorischen und optionalen Einschränkungen auf die notwendigen Daten, die für die Webanwendung benötigt werden. Die `getUserMedia()` Methode besitzt drei Parameter:

- eine Einschränkung *constraints object*,
- eine *Success callback* Funktion (Erfolgsmeldung), die den MediaStream übermittelt und
- eine *Error callback* Funktion (Fehlerrückmeldung), für das Übergeben der Fehlermeldung.

[26] http://chimera.labs.oreilly.com/books/1230000000545/ch18.html#GETUSERMEDIA

Abbildung 3.8: Die Abfrage im Browser für die Erlaubnis, um auf die entsprechende Hardware zugreifen zu können.

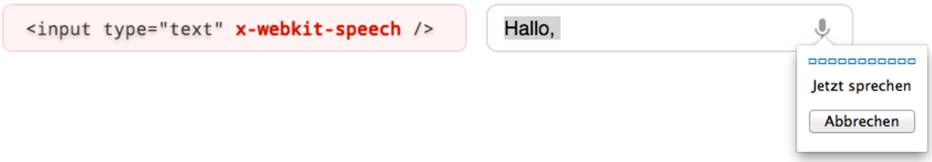

Abbildung 3.9: Anwendung des x-webkit-speech Attributes bei einem Input Elements.

Durch diese Methode wird auf die Webcam und auf das Mikrofon des Gerätes zugegriffen, wenn die Erlaubnis für den Zugang gegeben ist, die vom Webbrowser angefordert wird.

```
1  // getUserMedia( constraints, Success callback, Error Callback )
2  navigator.getUserMedia ({ video: true, audio: true },
3    // Success Callback
4    function(localMediaStream) {
5      var video = document.querySelector('video');
6      video.src = window.URL.createObjectURL(localMediaStream);
7      video.onloadedmetadata = function(e) {
8        // Do something with the video here.
9      };
10   },
11    // Error Callback
12    function(err) { console.log("The following error occured: " + err); }
13 );
```

Um auf Webcam und Mikrofon zugreifen zu können, müssen bei den *Contraints* beide mitübergeben werden. Durch die Bestätigung (Abb. 3.8) des Zugriffes auf die Beiden wird dann die *Success Callback* Funktion ausgeführt und ein Stream zurückübergeben, der auf Webcam und Mikrofon zugreift. Mit dem ankommenden Stream kann dann weitergearbeitet werden. Entsprechend des Verwendungszweckes wird der Stream dann mittels JavaScript manipuliert und verarbeitet.

3.4 Web Speech API

Die Web Speech API[27] ermöglicht einen Zugriff auf Sprachinput sowie auf eine Ausgabe von Text als Sprache, *text-to-speech output*. Diese Features sind grundsätzlich nicht vorhanden bei der Verwendung von Spracherkennungs- oder Screen Reader Software. Die Web Speech API kümmert sich um die Abwicklung zwischen Browser und Benutzer durch die Abfrage der Zugriffserlaubnis. Diese Zugriffsabfrage ist ähnlich wie die bei der `getUserMedia()` API, nur wird in diesem Fall der Zugriff auf das Mikrofon eingeschränkt. Die API umfasst zwei sehr komplexe Schnittstellen: das *SpeechRecognition Interface* und das *SpeechSynthesis Interface*.

Die Spezifikation der Web Speech API stellt fest, dass die API auf die gegebenen Implementierung der Spracherkennung und Synthese agnotisch ist und serverbasierte sowie clientbasierte Spracherkennung und Synthese unterstützt. Die API erlaubt zwei Arten von Spracherkennung: einmalige und kontinuierliche Erkennung. Bei der *einmaligen* Erkennung wird die Aufnahme des Stream an den Benutzer gebunden. Mit Beendigung des Sprechens des Benutzers wird auch die Aufnahme gestoppt. Bei der *kontinuierlichen* Erkennung ist, wie schon der Name sagt, die Aufnahme durchgehend, bis der Benutzer die Aufnahme unterbricht. Weiters ist auch noch erwähnenswert, dass die API ein *grammar object* erlaubt, welches die Regeln für eine definierte Sprache beschreiben.

Da schon vor der Entwicklung der Web Speech API Implementation für Spracherkennung, wie zum Beispiel das *x-webkit-speech* Attribut[28], existierten ist die Browserkompatibilität bei der Web Speech API (Abb. 3.10) noch nicht sehr ausgereift.

SpeechRecognition API

Die Methoden und Eigenschaften der SpeechRecognition API[30] umfassen eine breite Palette, die eine Weiterverarbeitung und Manipulation hervorragend anbietet. Um ein sogenanntes *recognizer* Objekt zu erstellen wird mittels der `speechRecognition()` Methode dieses Objekt instanziert.

```
1 var recognizer = new speechRecognition();
```

Folgende Methoden können dann auf dieses Objekt angewendet werden:

- `onstart`: erzeugt ein Callback beim Beginn der Spracherkennung und „hört zu“. Durch den Beginn des `recognition service` lauscht dieses Callback mit dem Zweck Sprache zu erkennen.

[27] Web Speech API: https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html

[28] Beispiel für die Verwendung des x-webkit-speech Attributes: http://slides.html5rocks.com/#speech-input

[30] SpeechRecognition Interface: https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html#speechreco-section

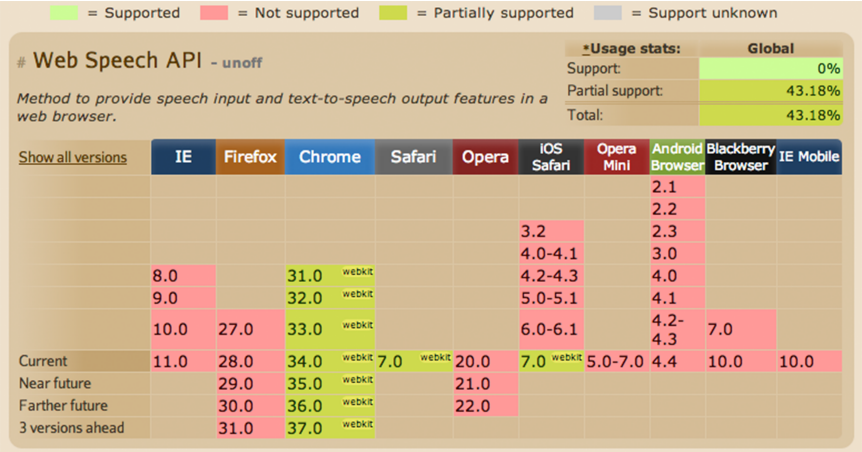

Abbildung 3.10: Browserkompatibilität[29] der Web Speech API.

- `onresult`: erzeugt einen Callback wenn ein Ergebnis durch die Spracherkennung festgestellt wurde. Hier muss das `SpeechRecognitionEven` Interface verwendet werden.
- `onerror`: erzeugt ein Callback wenn ein Fehler durch die Spracherkennung generiert wird. Hiert muss das `SpeechRecognitionError` Interface verwendet werden.
- `onend`: erzeugt ein Callback bei Beendigung der Spracherkennung. Dieses Event muss immer erstellt werden, wenn eine Session stoppt, unabhängig von der Ursache der Stoppung.

Zusätzlich zu diesen Methoden können noch folgende Einstellungen auf das `recognizer` Objekt angewendet werden:

- `continuous`: definiert die Art der Spracherkennung (einmalig und kontinuierlich)
- `lang`: definiert die Sprache für die Erkennung
- `interimResult`: definiert ob der Inputstream einstweilig untersucht werden soll. In diesem Fall kann das Ergebnis von dem `recognizer` Objekt durchgehend analysiert werden oder erst nach Beendigung der Aufnahme.

Durch das `recognizer` Objekt und den EventHandler `onresult` wird ein Objekt vom Type *SpeechRecognitionEvent* zurückgegeben. Mit diesem Objekt kann dann entsprechend weitergearbeitet werden. Der Zugriff auf die Daten erfolgt durch die Manipulation des `result[i]` Arrays:

```
1 record.onresult = function(event) {
2   if(event.results.length){
3     for (var i = event.resultIndex; i < event.results.length; ++i) {
```

[30] http://caniuse.com/#search=web%20speech%20api

```
4       var str = event.results[i][0].transcript;
5     // Play the video
6       if (userSaid(str, 'start')) { media.play(); }
7     // Stop the video
8       else if (userSaid(str, 'stop')) { media.pause(); }
9     }
10  }
11 };
```

`x-webkit-speech` Attribut

Wie schon erwähnt, existiert ein *x-webkit-speech* Attribut und wird bei dem Input Element angewendet (Abb. 3.9). Dieses Attribut kann auf HTML5 Input Element angewendet werden, dessen Typ *text, number, tel* oder *search* aufweist. Für Textfelder Textarea Elemente ist diese Attribut nicht anwendbar. Unter anderem ist dieses Attribut auch auf die Webbrowser Unterstützung angewiesen und wird mit einer Überprüfung abgeklärt.

```
1 if (document.createElement("input").webkitSpeech === undefined) {
2   alert("Speech input is not supported in your browser.");
3 }
```

Kapitel 4

Eigener Ansatz – Prototype

Die Motivation für die kombinierte Kontrolle webfähiger Elemente durch Handgesten und Sprachbefehle führt zu einem Protoypen, der auf eine HTML5 Präsentation angewendet wird. In diesem Kapitel wird der Ansatz für diesen Prototypen näher betrachtet und die Herangehensweise für den Protoyp selbst.

4.1 Anregung zum Konzept

Der Einsatz von Handgesten Erkennung ist aktuell in den verschiedensten Bereichen sehr stark eingebunden. Eines der bekanntesten Beispiele der Verwendung von Handgesten findet man im *Games* Bereich. Microsoft Kinect für die Xbox One oder 360, PlayStation Eye Camera oder die Kamera inkludiert in Nintendo Geräten sind alle mit unterschiedlichen Verarbeitungsmethoden ausgestattet, um Handgesten oder Bewegungen vor der Kamera zu verstehen. Das geeignetste Beispiel ist die Microsoft Kinect Kamera (siehe Verwandte Arbeiten 2.4).

Diese Art der Kommunikation mit dem Gerät ist eine sehr intuitive Umgangsform. Es läuft darauf hinaus, dass der Mensch mit dem System soweit in einer Kommunikation steht, dass eine direkte körperliche Verbindung zum Gerät nicht nötig ist, um Befehle ausführen zu können. Die Definition der verwendbaren Handgesten ist ein wichtiger Bereich in diesem Zusammenhang. Die Gesten sollten intuitiv, logisch und natürlich für den Menschen sein [2]. Werden Gesten eingebunden, welche gegen die natürlichen Instinkte arbeiten, kann dies zu Verwendungsproblemen führen. Eine intuitive Geste ist zum Beispiel das Winken vor der Kamera, um die Microsoft Kinect zu aktivieren. Ebenso sind Handbewegungen von Links nach Recht natürlich für das Umblättern von Seiten.

Bei den Sprachbefehlen ist es sehr ähnlich mit der Intuitivität. Für das Auslösen von Events sollten Kommandos verwendet werden, die eine Person auch im alltäglichen Gerauch verwenden würde. Angenommen man möchte

ein Video starten, so würde man dies mit den Worten „starten“, „beginnen“ oder auch „abspielen“ ausdrücken. Da die Spracherkennung auf einzelne Worte definiert werden kann, somit nicht zwingend auf vollständige Sätze, kann eine breite Palette an Ausdrücken erfasst werden.

Diese Kombination von Körper und Sprache ist ein sehr natürlich wirkender Umgang mit einem Endgerät. Der Mensch ist es gewohnt, sich durch Gesten und Sprache auszudrücken bzw. seine Absichten zu übermitteln. Somit ist die Verwendung von Gesten und Sprache für den Menschen natürlich. Die meisten Geräte, die mit solchen Systemen arbeiten, besitzen eine ensprechende Software inklusive eigener Kamera für die Handgesten Erkennung. Für den Protoypen dieser Arbeit wurde dieses System auf die Webebene gehoben. Hier werden mittels integrierter Webcam und Mikrofon die nötigen Daten erhoben und verarbeitet. Die Handgesten und Spracherkennung wird auf eine HTML5 Präsentation angewendet um die Events und Auswirkungen zu testen. Durch die Kombination sollen die Handgesten das Navigieren der Präsentation übernehmen und somit die Folien steuern. Die Sprachbefehle sind für die Manipulation von Media Elementen zuständig, wie zum Beispiel das Aktivieren einer Audiosequenze oder das Stoppen eines Videostreams. Die kombinierte Kontrolle soll es dem Benutzer ermöglichen eine Präsentation abzuhalten ohne das Gerät angreifen zu müssen. Somit wird der Benutzer, in einem definierten Abstand, vor dem Gerät stehen und die entsprechenden Befehle ausführen um durch die Präsentation zu führen.

4.2 Kombination von Handgesten und Sprachbefehlen

Für die Kombination (Abb. 4.1) der beiden Erkennungsvarianten müssen die entsprechenden Technologien (siehe oben 3) eingebunden werden um auf Webcam und Mikrofon zugreifen zu können. Der erhaltene Input muss dann anschließend analysiert und weiter verarbeitet werden. Die Technologien, die hier eingebunden werden, benötigen bestimmte Voraussetzungen um aktiviert werden zu können. Browserkompatibilität, Hardware-Bestandteile sowie die Grundstruktur der Präsentation sind unumgängliche Erfordernisse für dieses System. Es kann aber auch durch den Benutzer eine fehlerhafte Situation für das System entstehen, wenn der Zugriff auf die Hardware abgebrochen wird. Sollten die Erfordernisse nicht vorhanden sein, muss auf eine Alternative (siehe unten 4.3) umgestiegen werden.

Handgesten

Für die Präsentation wurden Handgesten vordefiniert, die später in einer Benutzerstudie auf Effektivität und Anwenderintuitivität geprüft werden. Die Definition dieser Handgesten beruht auf bereits vorhandenen Systemen, die

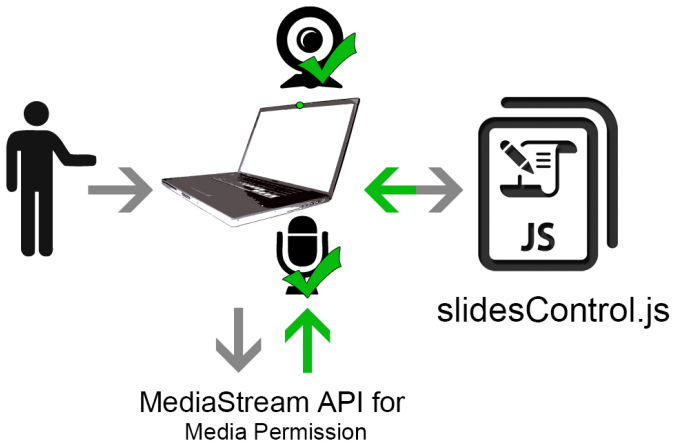

Abbildung 4.1: Grundlegender Ablauf des Systems

mit Handgesten arbeiten (Verwandte Arbeiten siehe Abschnitt 2.4). Neben diesen Vorlagen ist auch die Wischgesten Verarbeitung auf mobilen Endgeräten eine Quelle für das Festlegen der Handgesten gewesen. Die Wischgesten auf einem mobilen Endgerät verhalten sich ähnlich zu dem Konzept dieser Arbeit wie zum Beispiel das Navigieren über die unterschiedlichen Bildschirme. Für die Navigation der Präsentation wurden folgende Gesten festgelegt:

- RTL: *Rechts nach Links* - Bewegung zum Navigieren zur nächsten Folie,
- LTR: *Links nach Rechts* - Bewegung zum Navigieren zur vorherigen Folie,
- BTT: *Unten nach Oben* - Bewegung zum Navigieren zur darunterliegenden Folienebene,
- TTB: *Oben nach Unten* - Bewegung zum Navigieren zur darüberliegenden Folienebene,
- OA: *Overview aktivieren* mit zwei Händen von *Unten nach Oben* - Bewegung und
- OD: *Overview deaktiviern* mit zwei Händen von *Oben nach Unten* - Bewegung.

Die Analyse der Handgesten übernimmt ein JavaScript File, welches auf die Veränderung in den Bilddatenstream achtet und anschließend entsprechende Events aufruft um die Folien der Präsentation zu manipulieren.

Sprachbefehle

Bei den Sprachbefehlen ist es ganz ähnlich wie bei den Handgesten. Es gibt auch hier vordefinierte Werte, die als Befehle (Wörter) dem Benutzer dienen. Bei der Verwendung der Befehle werden diese Wörter überprüft und die

entsprechenden EventHandler führen die gesetzten Aktionen aus. Abhängig von den Befehlen werden die Media Elemente manipuliert. Die Befehle gelten für beide Variation von Media Elementen: Audio und Video. Folgende Worte können vom Benutzer verwendet werden:

- *start* und *play* für das Starten
- *stop* und *finish* für das Stoppen
- *reload* für erneutes Abspielen
- *pause* für eine Unterbrechung

Die Wörter können in einem Satz verwendet werden, da das System auf den gesamten gesprochenen Audiostream achtet und bei einer Übereinstimmung des Abgleiches wird der entsprechende Befehl ausgeführt.

Aktivierung und Verknüpfung der Handgesten mit Sprachbefehlen

Mit der Aktivierung der Webcam und des Mikrofons durch die Funktion *getUserMedia()* werden die Audio- und Videostreams für die Anwendung zugänglich. Die Analyse und Manipulation wird mittels JavaScript und Zugriff auf das Reveal.js Framework und die HTML5 Media API durchgeführt. Handgesten lösen ein Reveal.js Event aus und Sprachbefehle rufen die Media API für die Video und Audio Elemente auf. Die Handhabung und das Auslösen der nötigen Events wird in der Implementierung des Steuerungsscripts (siehe Abschnitt 5) genauer aufgezeigt. Ebenso wie die erforderlichen Technologien für Hardware Zugriffe. Durch diese Kombination soll der Redefluss des Benutzers unterstützt werden. Der Benutzer ist nicht abhängig von einer Fernbedienung und kann sich bis auf einen bestimmten Radius vom Gerät entfernen. Steht der Benutzer in der Zone, welche die Webcam erfasst, kann die Navigation durchgeführt werden. Abhängig von der Sprechlautstärke kann auch von einer etwas größeren Entfernung ein Sprachbefehl ausgeführt werden. Die entsprechenden Werte sind bei der Benutzerstudie und der Evaluierung (siehe Abschnitt 6) erfasst worden.

4.3 Alternative zur Steuerung

Betrachtet man den Prototypen, erkennt man, dass bestimmte Grundkenntnisse sowie Einstellungen vorhanden sein müssen. Sind diese Voraussetzung nicht gegeben, muss auf eine Alternative zurück gegriffen werden.

Reine Sprachsteuerung oder Gestenerkennung

Der Protoype greift auf Webcam und Mikrofon des Gerätes zu. Ist eines der beiden nicht vorhanden, so muss eine Ausweichlösung in Betracht gezogen werden. Je nach Situation kann dann durch die Benutzerbestätigung erkannnt werden, welcher Geräteanschluss nicht vorhanden ist. Anhand dieser

Information kann der Prototype entsprechend reagieren und die notwendigen EventHandler für die Aktionen des Protoypes einsetzten.

- Im Falle einer fehlenden Webcam: Wird die Anfrage für den Videostream durch den Benutzer abgebrochen, kann der Prototype mit der *Error Callback* Funktion (Fehlerrückmeldung) neue *EventHandler* setzen und entsprechend die Aktionen und Events neu behandeln, die durch Handgesten ausgelöst worden wären.
- Im Falle eines fehlenden Mikrofons: Bei Abbruch der Anfrage für den Audiostream kann ebenfalls durch die *Error Callback* Funktion festgelegt werden, welche neuen EventHandler nun reagieren müssen, wenn eine Sprachbefehl erwartet worden wäre.

Keyboard und Mouse

Die Verwendung von Keyboard (Tastatur) und/oder Mouse (Computermaus) kommt in Frage, wenn weder die entsprechende Hardware für das System vorhanden ist, oder die Zugriffsberechtigung nicht erteilt wurde. In dieser Situation kann einfach mit der Mouse via grafischer Darstellung (auf der Präsentationsoberfläche im rechten unterem Eck) die Navigation der Präsentation gesteuert werden. Auch mittels Keyboard ist die Präsentation sehr einfach mit den Pfeiltasten zu steuern. Optional wird bei Media Elementen die Leertaste verwendet.

Kapitel 5

Implementierung

Für die Umsetzung des Prototypen müssen bestimmte Rahmenbedingungen erfüllt werden. Um die Anwendung entsprechend starten zu können muss diese zumindest auf einem lokalen Server gestartet werden. Für den Prototypen dieser Arbeit wurde *Node.js*[1] und *Grunt.js*[2] verwendet.

Um alle weiteren nötigen Technologien einbinden und verwenden zu können, muss bzw. sollte die Anwendung in der aktuellsten Version des Google Chrome[3] Webbrowser gestartet werden. Ist kein Google Chrome Browser vorhanden kann auf die aktuellste Version von Mozilla Firefox[4] ausgewichen werden. Ansonsten müssen Webbrowser gewählt werden, die folgende Technologien unterstützen: WebRTC API und Web Speech API. Die Anwendung muss über einen Webserver laufen, damit der Zugriff auf die Hardware des Gerätes abgefragt werden kann. Für die vollständige Verwendung aller Funktionalitäten der Anwendung müssen für das Gerät eine Webcam sowie ein Mikrofon bereitgestellt werden. Webcam sowie Mikrofon können sowohl interne als auch externe Hardware angesprochen werden. Sind diese Voraussetzungen erfüllt, kann die Anwendung problemlos gestartet und bedient werden, ansonsten muss auf Ausweichmöglichkeiten zurückgegriffen werden. Die vollständige Implementierung des Prototypen umfasst die HTML5 Struktur der Präsentation und die Kontrolle durch JavaScript Zugriffe. Auch die notwendigen Schnittstellen müssen entsprechend eingebunden werden um mit den entstehenden Objekten weiterarbeiten zu können. Somit kann die HTML5 Präsentation manipuliert und der Ablauf gesteuert werden.

[1] http://nodejs.org/

[2] http://gruntjs.com/

[3] Google Chrome Version am 20. April 2014: Version 34.0.1847.116 https://www.google.com/intl/en/chrome/browser/

[4] Mozilla Firefox Version am 20. April 2014: Version 28.0 https://support.mozilla.org/de/products/firefox

5.1 Struktur der HTML5 Präsentation

Die Präsentation benötigt eine vordefinierte Struktur damit die Inhalte korrekt angezeigt werden. HTML5 (siehe Kapitel Technologien 3.1) gibt die Struktur vor und ermöglicht dem Benutzer in definierten Bereich Inhalte einfügen zu können, wie Text, Bilder, Audio und weitere Präsentationselemente. In Kombination mit Reveal.js wird die Struktur durch das Reveal.js-Layout für die Präsentation ein weiteres Mal vordefiniert und beschrieben, um später das typische Präsentationsverhalten zu ermöglich.

Erforderliche HTML5 Elemente

Für den Inhalt der Folien und Media Dateien werden bestimmte Elemente genommen. Die nötige Grundstruktur ist mit folgenden Elementen definiert:

- <html> Element,
- <body> Element,
- Elemente mit sprechenden *Classen* für Reveal.js Einbindung und
- <section> Elemente, welche einen Absatz in einem Dokument definieren und hier den Bereich einer Folie.

Mit diesen Hauptelementen (Prog. 5.3) können weitere Inhalte eingebunden werden. Überschriften und Texte sind einfach einzubetten in den Section Elementen. Hier werden die üblichen HTML Elemente verwendet, wie zum Beispiel die `<h1>,<p>` oder `<span>` Elemente. Für die Verwendung von Audio, Video und Bild-Dateien werden eigene Elemente genommen. Für die Einbindung von Code Segmenten und Auflistungen bietet HTML5 entsprechende Elemente an, welche von Reveal.js mit vordefinierte Styles in der HTML5 Präsentation dargestellt werden. Somit sind unter anderem folgende Elemente[5] für die Einbindung verwendbar:

- <pre> und <code> Elemente für Code Segmente,
- , <ol> und <li> Elemente für geordnete und ungeordnete Listen,
- <audio> und <video> Elemente inklusive des
- <source> Elementes für Audio und Video Daten,
- <img> Elemente für Bilddaten.

Bei der Verwendung von Audio und Video Elementen müssen diese entsprechend definiert werden. Mittels einer bestimmten *id* (Prog. 5.1) können diese Elemente dann später durch das Kontroll-Script angesprochen werden und manipuliert werden. Somit können Audio und Video Elemente auf einer Folie entdeckt werden und somit auch gestartet, gestoppt oder ähnlich behandelt werden.

[5] Liste der HTML Elemente: https://developer.mozilla.org/en/docs/Web/Guide/HTML/HTML5/HTML5_element_list

Programm 5.1: Fixe Struktur für Audio und Video Elemente auf Grund des *id* Attributes.

```
1 <!-- id "media0" for the first media part -->
2 <audio id="media0" controls>
3 <!-- id "media1" for the second media part -->
4 <video id="media1" width="620" height="440" controls>
```

Bei der Untersuchung einer Folie wird überprüft, ob sich eine derartige *id* bei einem Media Element befindet. Weitere Schritte werden dann mit dem Kontroll-Script behandelt.

Struktur der Präsentation

Durch die Kombination der HTML5 Elementen und der benötigten Reveal.js Markup Hierarchie wird eine Struktur für die Präsentation vorgegeben, die nicht verändert werden sollte. Reveal.js (siehe Kapitel Technologien 3.2) stellt durch die Initialisierung Layout, Styles und Animationen zur Verfügung. Für die Anwendung auf die HTML Struktur muss eine festgelegte Markup Hierarchie (Prog. 5.2) eingehalten werden: `<div class="reveal">`, `<div class="slides">` und Section Element. Das Section Element kann so oft wie benötigt wiederholt werden um alle Folien darstellen zu können. Mit den JavaScript API Aufrufen kann mit dem Kontroll-Script später die Navigation der Präsentation behandelt werden. Reveal.js besitzt eigene EventListener, die für den Start der Navigation und die Änderung der Folien zuständig sind. Diese Gegebenheiten müssen bei der Erstellung der Präsentation berücksichtigt werden sowie bei den Aufrufen durch das Kontroll-Script für die Handgesten und Sprachbefehle. Ohne die Einhaltung dieser Hierarchie kann weder das Layout noch das Bewegen der Folien aktiviert werden. Die Einhaltung dieser festgelegten Struktur (Prog. 5.3) für den Prototypen und die Einbindung der notwendigen Elemente inklusive sprechender *id's* für die Media Elemente ist wichtig. Mit diesem Aufbau kann anschließend das Kontroll-Script erkennen, wo sich Audio oder Video Elemente befinden. Das Navigieren wird mit der Einteilung Folieweise abgegrenzt und bei einer Handgeste kann entprechend mit den Folien gearbeitet werden. Daher ist die Struktur der Präsentation ausschlaggebend für den späteren Erfolg betreffend Handgesten und Sprachbefehlen.

5.2 Zugriff auf Media Stream

Für die Verwendung von Video und Audio Streams wird eine Webschnittstelle [4] benötigt. Der Zugriff erfolgt mittels JavaScript Aufrufen und muss vom Benutzer zugelassen werden. Bei der Abfrage können durch unterschied-

Programm 5.2: Fixe Vorgabe für die Markup Hierarchie von Reveal.js.

```
<div class="reveal">
  <div class="slides">
    <section>Single Horizontal Slide</section>
    <section>
      <section>Vertical Slide 1</section>
      <section>Vertical Slide 2</section>
    </section>
  </div>
</div>
```

liche Wege Audio und Video Stream aktiviert werden. Für die benötigten Bereiche wird dieser dann entsprechend eingesetzt und verwendet.

5.2.1 MediaStream API

Um die gewünschten Daten Streams zu erhalten wird von WebRTC [1] die MediaStream API verwendet. WebRTC bietet noch zwei weitere APIs (siehe Kapitel Technische Grundlagen 3.3.1) an, diese werden aber für den Prototypen nicht benötigt und daher wird auf die Applikation auf die MediaStream API [5] beschränkt. Mittels des *getUserMedia()* Aufrufes kann Audio und/oder Video angefordert werden. Da bei der Steuerung der Präsentation Handgesten und Sprachbefehle verwendet werden, werden Audio und Video Inputstreams benötigt. Da die Handgesten über den Videostream in Echtzeit erkannt werden wird das Videosignal durchgehend in Anspruch genommen. Bei dem Audiosignal wird nur bei Folien mit Media Elementen der Stream verwendet. Da jedoch eine ständige Aktivierung und Deaktivierung des Mikrofones für einzelne Folien ungünstig ist, wird das Mikrofon von Beginn an mit der Webcam aktiviert. Um mit beiden Streams getrennt arbeiten zu können werden diese auch getrennt aufgerufen. Der Videostream wird mit *getUserMedia()* angefragt und der Audiostream durch die Web Speech API mit dem *webkitSpeechRecognition()* Objekt. Bei dem Aufruf von *getUserMedia()* wird das *Navigator* Interface[6] verwendet. Dieses Interface stellt den Zustand und die Identität des *user agents* dar. Dies ermöglicht einen Zugriff mittels Scripte um weitere Aktivitäten aus zulösen. Werden unterschiedliche Webbrowser für die Applikation verwendet, sollten vorab für den *getUserMedia()* Aufruf alle Browser Prefixes auf das *navigator.getUserMedia* Objekt angewendet werden. In Falle des Prototypen (Prog. 5.4) dieser Arbeit werden Chrome und Firefox unterstützt und müssen daher vorab sicherheitshalber auf das Navigator Interface geprüft werden.

Bei der Anforderung des Videostream (Prog. 5.5) wird durch den Browser der Benutzer gebeten die Webcam zu aktivieren und dadurch erhält die

[6] https://developer.mozilla.org/en-US/docs/Web/API/Navigator

Programm 5.3: Struktur der HTML5 Präsentation mit horizontalen und vertikalen Folien. <section> Elemente werden als Inhaltsbereich für Folien verwendet.

```
<div class="preview">
  <!-- reveal, slides classes for presentation style -->
  <div class="reveal">
    <div class="slides">
    <!-- content in section element is one slide -->
      <section>
        <h1 style="font-size: 2.3em">HTML5 Presentation</h1>
        <h3>controlled by hand gesture and speech recognition</h3>
        <div>Elisabeth Baumgartner<br>FH Hagenberg</div>
      </section>
      <section> <!-- Content for the audio part -->
        <h2>Audio Part</h2>
        <audio id="media0" controls>
          <source src="media/audio.mp3" type="audio/mpeg">
          ...
          Your browser does not support this audio format.
        </audio>
      </section>
      <section> <!-- Content for the video part -->
        <h2>Video Part</h2>
        <video id="media1" width="620" height="440" controls>
          <source src="media/video1.mp4" type="video/mp4">
          ...
          Your browser does not support the video tag.
        </video>
      </section>
      <section> <!-- Content for code part -->
        <h2>Some code</h2>
        <pre>
          <code data-trim>
            if (navigator.getUserMedia === undefined) {
              console.log("No support for getUserMedia");
            }
          </code>
        </pre>
      </section>
      <section> <!-- Content for slide -->
        <section> <!-- Content for vertical slide one -->
          <h2>Vertical slide</h2>
          <div>A wonderful serenity ...</div>
        </section> <!-- Content for vertical slide two -->
        <section data-background-image="https://images/background.jpg">
          <h2>Background Image</h2>
        </section>
      </section>
    </div>
  </div>
</div>
```

Programm 5.4: Überprüfung auf Existenz von *navigator.getUserMedia()* für Chrome und Firefox.

```
1 navigator.getUserMedia =
2   ( navigator.getUserMedia || navigator.webkitGetUserMedia ||
3     navigator.mozGetUserMedia || navigator.msGetUserMedia );
```

Applikation einen Zugriff auf den Videostream. Bei einer erfolgreichen Anfrage wird der Stream an das *webcamSwiperStream* Objekt übergeben, welches als Hilfsobjekt für die Übergabe an das Video Elements dient. Durch die Erstellung und Einbindung in die HTML Struktur eines Video Elementes kann anschließend das Video mit dem Stream befüllt werden und ein EventListener erzeugt werden. Dieser ist zuständig für die Aktivierung der Funktionalität der Gestenerkennung.

Aufgrund der unterschiedlichen Browser wird auch vorab geprüft ob es sich bei dem Video Element um einen Firefox oder ein Chrome Webbrowser handelt. Dadurch kann der Stream den Browsern entsprechend behandelt werden. Ist der Stream an das Video Element übergeben wird der EventListener für die Gestenerkennung dem Element hinzugefügt. Ist das Video vollständig an das Video Element übergeben wird durch den EventListener die Funktion *gestureRecognition()* aufgerufen. Dadurch wird die Analyse für Handgesten und Manipulation der Präsentation gestartet.

Sollte der Stream für das Video nicht gegeben sein, wird die *Error Callback* Funktion (Fehlerrückmeldung) ausgelöst und dem Benutzer mitgeteilt, dass die Anfrage für den Stream nicht funktioniert hat. Die *errorCallback* Funktion wird aufgerufen, wenn ein Fehlverhalten in der Anwendung festgestellt wurde. Das betrifft einen misslungenen Verbindungsaufbau zu den Hardware Elementen oder auch ein Problem bei der Verbindung zu den nötigen Web APIs. Ein Fehlverhalten kann auch vom Benutzer ausgelöst werden, wenn er den Zugriff auf die Hardware nicht zulässt. Somit kann bei einem Fehlverhalten von der Applikation eingegriffen werden und ein erneuter Versuch gestartet werden.

5.2.2 SpeechRecognition Objekt

Mit der Web Speech API (siehe Kapitel 3.4) wird ein eigenes Objekt erstellt, das den Zugriff auf den Audiostream aktiviert und anschließend damit gearbeitet. Da es sich um eine reine Aufnahmesituation des Audiostreams handelt wird nur die SpeechRecognition API der Web Speech API genutzt. Entsprechend der Folien, auf denen sich Media Elemente befinden, wie zum Beispiel Audio oder Video Daten wird der Audiostream zu diesem Zeitpunkt aufgenommen und analysiert. Das Starten und Enden einer Aufnahme sowie die Analyse wird von `EventListener` übernommen. Wie

Programm 5.5: Aufruf von `getUserMedia()` und die Behandlung des Streams sowie Übergabe an die Gestenerkennung Funktion.

```
/** user media data for video has to be true
 *   video for the hand gesture
 *   @params constraints, successCallback, errorCallback
 */
navigator.getUserMedia({video: true, audio: false}, function (stream) {
   // successCallback function
   window.webcamSwiperStream = stream;

   // Create a video element and set its source to the stream from the webcam
   videoElement = document.createElement("video");
   videoElement.style.display = "none";
   document.getElementsByTagName("body")[0].appendChild(videoElement);
   if (window.URL === undefined) {
       window.URL = window.webkitURL;
   }
   if (videoElement.mozSrcObject !== undefined) {
       videoElement.mozSrcObject = stream;
   } else {
       videoElement.src = window.URL.createObjectURL(stream);
   }
   videoElement.play();

   // Wait for the video element to initialize
   videoElement.addEventListener("loadeddata", gestureRecognition);
}, function(err) {// errorCallback
     console.log('Something went wrong in getUserMedia');
});
```

schon erwähnt, wird der Audiostream separat angefordert und nicht durch den `getUserMedia()` Aufruf. Dadurch wird das Audiosignal getrennt aufgenommen und kann für die Sprachbefehle nur bei den benötigten Folien analysiert werden. Starten und Stoppen der Aufnahme beeinflusst somit nicht das Videosignal. Die Einstellungen (Prog. 5.6) für den Audiostream werden direkt bei dem Aufruf für das webSpeechRecognition Objekt definiert. Dabei wird die Sprache (`record.lang`) bestimmt, ob die Aufnahme durchgehend ist (`record.continuous`) und ob auf Zwischenergebnisse (`record.interimResults`) Rücksicht genommen werden soll. Mit diesen Einstellungen kann dann mit dem webSpeechRecognition Objekt gearbeitet werden, denn diese sind ausschlaggebend für den Vergleich von Sprachbefehlen mit den gesprochenen Wörtern des Benutzers.

Programm 5.6: Erzeugung des webSpeechRecognition Objektes für die Sprachbefehl Analyse.

```
1 // add webSpeechRecognition ————————————————
2 if ('webkitSpeechRecognition' in window) {
3      record = new webkitSpeechRecognition();
4      record.continuous = true;
5      record.interimResults = false;
6      record.lang = 'en';
7 } else {
8      console.log("Problem with the WebSpeech API");
9 }
```

5.3 Script für Steuerung der Präsentation

Das Steuern der Präsentation wurde mit JavaScript implementiert und übernimmt auch die Aufrufe an die Hardware Komponenten wie Webcam und Mikrofon. Die beiden Bereiche für Handgesten und Sprachbefehle sind übersichtlich aufgeteilt und werden mit dem Beginn der Aufnahme von Video und Audiostream aktiv. Um die Kombination der beiden zu ermöglichen müssen bei bestimmten Punkten die jeweiligen Aufrufe für die Analyse getätigt werden. Durch das Aufteilen des Streams in separate Audio und Video Objekte kann dann bei den entsprechenden Punkten parallel gearbeitet werden und es muss bei dem Audiosignal nicht auf eine ständige Analyse von Befehlen geachtet werden.

Wie der Ablauf zwischen den beiden Erkennungssystemen funktioniert und wie die Zusammenarbeit mit der Applikation abläuft wird in dem nachfolgenden Bereich erklärt (Abb. 5.1).

5.3.1 Handgesten Analyse

Um das Navigieren mit einer Handgeste auszulösen muss diese auch vorab erkannt werden. Nach dem Auslösen des *EventListeners* für die Gestenerkennung wird der Videostream mit der Funktion *gestureRecognition()* vorbereitet für die Analyse ob eine Handgeste getätigt worden ist. Mit Vorbereitung ist hier die Einstellung und Initialisierung des Video Elements und des Videostreams gemeint. Wurde das Video Element initialisiert, wird die Auflösung der Webcam festgestellt. Sollte bei der Auflösung schon ein Fehler kommen (eine fehlerhafte Auflösungssituation), muss rechtzeitig abgebrochen werden um das Videosignal noch mal neu zu untersuchen. Ist das Signal vorbereitet und zur weiteren Verwendung bereit wird der nächste Schritt für die Analyse eingeleitet. Um den Videostream analysieren zu können muss dieser in einem bestimmten Intervall mit der `analyseCurrentFrame()` untersucht werden:

Tabelle 5.1: Auswirkung der Lichtsituation auf den Pixeländerung und den Frame-Schwellenwert.

Lichtlevel	*PIXEL_CHANGE _THRESHOLD*	*FRAME _THRESHOLD*
zwischen 0 und 1	25	3000
zwischen 1 und 3	28	6000
kleiner als 0 und größer als 3	30	15000

```
1 // every 1000/25th of a second, sample/analyze the video stream
2 window.webcamSwiperInterval = setInterval(analyseCurrentFrame, 1000/25);
```

Funktion `analyseCurrentFrame()`

Bei der Implementierung der Funktion `analyseCurrentFrame()` wurden Schritte des Projektes „Webcam-Swiper“ in Betracht gezogen (siehe Kapitel Verwandte Arbeiten 2.1.1). Bei dieser Funktion wird der Videostream von den Farbwerten befreit. Für das Entfernen der Farbwerte wird die Funktion `deSaturate(videoElement)` angewendet und es wird ein Bild mit Graustufen erstellt und als `currentImage` hinterlegt. Dadurch kann die Funktion `getLightLevel(currentImage)` aufgerufen werden und der Zustand der Helligkeit bzw. der Lichtwert für die Applikation in dem Raum ermittelt werden. Abhängig von der Lichtsituation werden dann entsprechend Werte (Tab. 5.1) definiert.

PIXEL_CHANGE_THRESHOLD definiert den Schwellwert für geänderte Pixel in einem Frame. Dieser Wert ist für die spätere Überprüfung von einer Bewegung im Bild ausschlaggebend. Abhängig von der Lichtsituation muss der Schwellwert entsprechend festgelegt werden. Der Schwellwert für Pixeländerung im Bild wird später als Richtwert in der Funktion `getMotionWeight()` benötigt. FRAME_THRESHOLD definiert den Schwellwert für einen durchschnittlichen Frame und wird bei der Überprüfung für die Richtung der Handgeste benötigt. Für die *Frame Scan Rate*[7] wird ebenfalls eine Wert festgelegt, sollte sich die Framerate außerhalb des vorab definierten Wertes bewegen. Die Analyse wird mitgestoppt und würde es bei der Framerate zu einem höheren Wert kommen wird die Analyse an die erhöhte Zeit angepasst. Dadurch werden alle nötigen Einstellungen für das Videosignal übernommen und es kann mit der Beobachtung nach einer Bewegung begonnen werden. Die `getMotionWeight()` (Prog. 5.7) vergleicht zwei Bilddaten die hintereinander aufgenommen wurden und ermit-

[7] Die Häufigkeit der Untersuchung eines Frames.

Programm 5.7: Aufruf und Funktion `getMotionWeight()`

```
1 // Map the pixels that are changing - call getMotionWeight function
2 currentWeight = getMotionWeight(previousImageData, currentImageData);
3
4 /**
5  * getMotionWeight(previous, current)
6  */
7 function getMotionWeight (previous, current) {
8    var motionWeight = 0;
9    var previousData = previous.data;
10   var currentData = current.data;
11   var dataLength = previousData.length;
12   for (var i = 0; i < dataLength; i += 4) {
13      if (Math.abs(previousData[i] - currentData[i]) >
       PIXEL_CHANGE_THRESHOLD) {
14          motionWeight += ((i / 4) \% canvasWidth) - (canvasWidth / 2);
15      }
16   }
17   return motionWeight;
18 }
```

telt einen Bewegungswert `motionWeight` der angibt, ob und wieviele Pixel sich in den Bildern geändert haben. Der erhaltene Wert durch die Funktion `getMotionWeight()` ist später ausschlaggebend für die Erkennung von Handgesten und deren weiteren Auswirkungen auf die Präsentation. Beide Bilddaten `previousImageData` und `currentImageData` werden deren Länge entsprechend voneinander subtrahiert. Um negative Werte zu vermeiden wird mit der `Math.abs()` das Ergebnis beeinflusst. Der Unterschied, der bei der Berechnung zwischen *previousImageData* und `currentImageData` entsteht, wird mit dem Schwellwert `PIXEL_CHANGE_THRESHOLD` verglichen. Liegt das Ergebnis außerhalb des Schwellwertes wird der Wert für `motionWeight` berechnet, anderenfalls werden die nächsten Pixel der beiden Bilder gegenübergestellt. Der Wert `motionWeight` setzt sich aus den Pixeln zusammen, die bei dem Vergleich der beiden Bilder unterschiedlich sind.

Um zwischen vertikalen und horziontalen Handgesten unterscheiden zu können werden vor der Handgesten Analyse Funktion `analyseMovement()` Kalkulationen für die Errechnung von vertikalen Bildänderungen durchgeführt. Dabei werden die Pixel des Bildes untersucht und bei der Überschreitung von dem festgelegten Schwellenwert `thresh = 150` wird ein Objekt mit `x, y` und `d` Werten errechnet und befüllt, welches auf eine vertikale Geste hinweist. Dieses `down` Objekt enthält Werte, die auf Gesten von oben nach unten und umgekehrt schließen lassen.

Funktion `analyseMovement()`

Nach der Ermittlung des Bewegungswertes und der Kalkulation für vertikale Handgesten kann mit der Erkennung von Gesten begonnen werden. Der Ablauf (Abb. 5.1) von einer Funktion zur nächsten ist von den Ergebnissen der vorherigen Funktion abhängig. Die Funktion `analyseMovement()` liefert die Befehle für die Präsentation bei dem Entdeckten von Handgesten sowie dem Erkennen von sprachgesteuerten Elementen. Für die Erfassung von Handgesten werden vorab ein paar Einstellungen direkt in der Funktion geregelt:

- `isActive`: Ein boolscher Wert, der den Zustand für die Aktivierung der direkten Analyse beschreibt. Ist der Wert auf *wahr* gesetzt so darf davon ausgegangen werden, dass Handgesten vor der Webcam stattfinden werden. Wird der Wert auf *falsch* gesetzt, ist die Applikation dabei ein Event auf Grund einer Handgeste auszuführen.
- `remainingFrames`: Wert für Anzahl der erlaubten Nummer von aufeinander folgenden Frames, die analysiert werden. Dieser Wert durch die `isActive` Kontrolle bei jeder Analyse entsprechend gesetzt.
- `originalWeight`: erhält den Wert, der durch die Berechnung von Bewegung im Bild entsteht `currentWeight`.
- `wasDown`: erhält das `down` Objekt, welches für vertikale Handgesten errechnet wurde.

Neben den Einstellungen werden auch noch die Werte für die Bewegungsrichtung in einem Bild festgestellt. Dabei werden die Werte aus den beiden `wasDown` und `down` Objekten herangezogen. Diese Werte der Bewegungsrichtung sind dann für den Vergleich mit dem Frame-Schwellwert nötig. Mit den fertigen Einstellungen kann anschließend der Vergleich von `currentWeight` und `FRAME_THRESHOLD` sowie Bewegungsrichtung durchgeführt werden. Es gibt jeweils zwei Handgesten für horizontales und vertikales Navigieren. Bei einer horizontalen Handgeste handelt es sich um die beiden Bewegung von Links nach Rechts und umgekehrt. Für die beiden Handgesten werden folgende Werte (Prog. 5.8) miteinander verglichen:

- *currentWeight* kleiner als der negative `FRAME_THRESHOLD` Schwellwert: eine Bewegung von Links nach Rechts. Es wird die nächste Folie angezeigt.
- *currentWeight* größer als der `FRAME_THRESHOLD` Schwellwert: eine Bewegung von Rechts nach Links. Es wird die vorherige Folie angezeigt.

Wird eine vertikale Handgeste vor der Webcam erzeugt, so muss mit den vorab erstellten `down` und *wasDone* Objekten der `FRAME_THRESHOLD` (Prog. 5.9) verglichen werden. Werden die entsprechenden Bewegungen durch den Benutzer erkannt, wird die Präsentation manipuliert und zur gewünschten Folie navigiert. Da somit ein Folienwechsel erfolgt, muss der Inhalt dieser Folie untersucht werden, ob sich ein Media Element darin befindet. Mit der Funk-

Tabelle 5.2: Werte der Variablen `currentWeight` und `FRAME_THRESHOLD` während einer Präsentation. Es wurden Handgesten im horizontalen Bereich angewendet und die Applikation hat entsprechend darauf reagiert. `FRAME_THRESHOLD` hat kaum eine Wertänderung, da das Lichtverhältnis im Raum nicht geändert wurde.

`currentWeight`	`FRAME_THRESHOLD`	*ausgeführte Aktion*
-90061	15000	nächste Folie
33654	15000	vorherige Folie
-52975	15000	nächste Folie
-34943	15000	nächste Folie
-24296	15000	nächste Folie
20123	15000	vorherige Folie

Programm 5.8: Auszug aus der Funktion `analyseMovement()`. In diesem Codesegment wird die Prüfung auf eine Rechts-nach-Links und umgekehrte Bewegung dargestellt inklusive des nachfolgenden Aufrufen der Funktion *checkForMedia()* für Audio und Video Elemente auf einer Folie.

```
1  // movement from right to left
2  if (originalWeight > 0) {
3       if (currentWeight < -FRAME_THRESHOLD) {
4            isActive = false;
5            Reveal.navigateRight();
6            //check if there is an media element in section
7            checkForMedia();
8       }
9  } else {
10      // movement from left to right
11      if (currentWeight > FRAME_THRESHOLD) {
12           isActive = false;
13           Reveal.navigateLeft();
14           //check if there is an media element in section
15           checkForMedia();
16      }
17 }
```

tion `checkForMedia()` (Prog. 5.10) wird der gesamte Inhalt einer Folie, also der Bereich in dem Section Element, untersucht und behandelt somit die sprachgesteuerten Elemente.

Nachfolgendes Beispiel (Tab. 5.2), während einer Präsentation mit einem Benutzer, zeigt die Werte für die beiden Variablen an und deren Auswirkung auf die Präsentation.

Programm 5.9: Auszug aus der Funktion `analyseMovement()` mit den Einstellungen für die Berechnung der Richtungswerte. In diesem Codesegment wird die Prüfung auf eine Oben-nach-Unten und umgekehrte Bewegung dargestellt inklusive des nachfolgenden Aufrufen der Funktion `checkForMedia()` für Audio und Video Elemente auf einer Folie.

```
1  // detection of down movement
2  avg = 0.9 * avg + 0.1 + down.d;
3  var dx = down.x - wasDown.x;
4  var dy = down.y - wasDown.y;
5  var dirX = Math.abs(dy) < Math.abs(dx);
6
7  ...
8
9  // movement from down to top
10 if (dy > FRAME_THRESHOLD && !dirX) {
11      isActive = false;
12      Reveal.navigateUp();
13      checkForMedia();
14 } // movement from top to down
15 else if (dy < -FRAME_THRESHOLD && !dirX) {
16      isActive = false;
17      Reveal.navigateDown();
18      checkForMedia();
19 }
```

5.3.2 Sprachbefehl Analyse

Das Erkennen von Sprachbefehlen für die definierten HTML Elemente wird mit der Funktion `checkForMedia()` (Prog. 5.10) aktiviert. Der Aufruf dieser Funktion wird bei Navigation zur aktuellen Folie getätigt und somit wird jede Folie nur zum Zeitpunkt der Anzeige auf Media Elemente untersucht. Mit `checkForMedia()` wird der Inhalt eines Section Elementes der Klasse „present“ genauer betrachtet. Es wird auf zwei HTML5 Elemente geachtet: Audio und Video. Diese können mittels `childElements.localName` erkannt werden und deren *id* wird dann weiter an die Funktion `speechCommands()` gegeben. Somit werden die Sprachbefehle nur auf das HTML Element mit der entsprechenden *id* angewendet. Wurde ein Media Element erkannt wird die *id* weitergeben um somit alle Kommandos auf das Element ausführen zu können. Der Benutz muss zwar die vordefinierten Kommandos verwenden, kann aber mit mehreren Wörtern den gleichen Effekt erzielen. So sind zum Beispiel für das Abspielen eines Media Elements folgende Begriffe als Kommandos zugelassen: start und play. Die Befehle können in einem Satz eingebunden werden und somit in den Präsentationsfluss eingearbeitet werden. Befehle zum Pausieren oder Stoppen werden auch während des Abspielens des Media Elements erkannt. Für die Analyse der Kommandos müssen

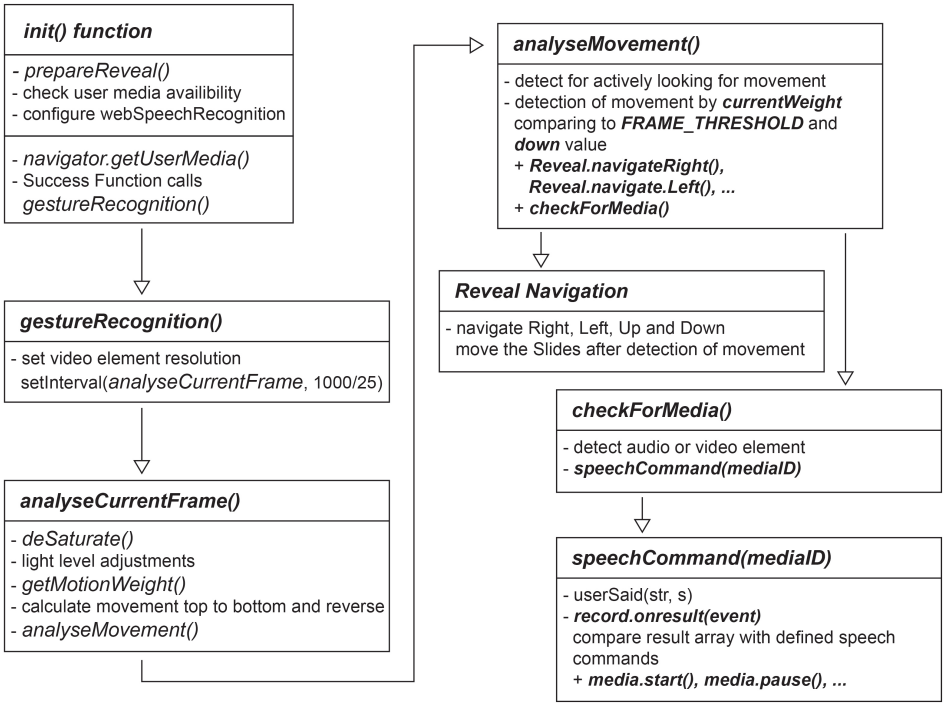

Abbildung 5.1: Ablauf und Zusammenhang zwischen den Funktionen des Konzeptes.

ebenfalls vorab Einstellungen getroffen werden. Bei der Spracherkennung sind folgende Werte festzulegen:

- `record.continuous`: Ein boolscher Wert, der auf *wahr* gesetzt wird, damit ab Aufnahme der Sprache alle Wörter untersucht werden. Würde *falsch* als Wert gesetzt werden, würde ab dem ersten erkannten Kommando die Aufnahme abbrechen und nachfolgende Kommandos ignoriert werden.
- `record.interimResults`: Ein boolscher Wert, der auf *wahr* gesetzt wird, da ansonsten Zwischenergebnisse zurück gegeben werden, die durch schlechte Aussprache entstanden sind. Wird der Wert auf *falsch* gesetzt, kommen alle Zwischenergebnisse zurück, was bei einer Textausgabe der aufgenommen Sprache oft zu schlechten Ergebnissen führt.
- `record.lang`: Wert, der angibt welche Sprache gesprochen wird.

Mit den Einstellungen können die Aussagen des Benutzers aufgenommen werden. Die Aufnahme erfolgt über das `webkitSpeechRecognition` Objekt und wird durch die Funktionen gesteuert, die die SpeechRecognition API zur Verfügung stellt. Mit dem Aufruf `onresult` in der Funktion *speechCommands(id)* (Prog. 5.11) wird die Aufnahme als Array an die Applikation

Programm 5.10: Funktion für die Überprüfung ob ein Video oder Audio Element auf der aktuellen Folie zu erkennen ist und Aufruf der Funktion für die Sprachanalyse.

```
1 /**
2  * check the active slide for audio or video files
3  */
4 function checkForMedia(){
5     var checkElement = document.getElementsByTagName('section');
6     for(var i= 0; i < checkElement.length; i++){
7         if(checkElement[i].className == 'present'){
8             var childElements = checkElement[i].children;
9             for(var j=0; j < childElements.length; j++){
10                if(childElements[j].localName == 'audio' ||
      childElements[j].localName == 'video'){
11                    mediaID = childElements[j].id;
12                    speechCommands(mediaID);
13                    return;
14                }
15            }
16        }
17    }
18 }
```

übergeben und kann damit untersucht werden. Das zurückgegeben `event` Objekt enthält folgende Daten:

- `results[i]`: Ein Array mit den erkannten Objekten, wobei jedes Element als erkanntes Wort zählt.
- `resultsIndex`: Der aktuelle Index des Recognition-Ergebisses.
- `results[i].isFinal`: Ein boolscher Wert, der angibt ob ein Ergebnis fertig ist oder als Zwischenergebnis zählt.
- `results[i][j]`: Ein 2D Array mit alternativ anerkannten Wörtern als Resultat. Das erste Element gilt hier als das wahrscheinlichste Wort.
- `results[i][j].transcript`: Die Textdarstellung der erkannten Wörter.
- `results[i][j].confidence`: Die Wahrscheinlichkeit für die Richtigkeit des Ergebnisses

Mit dem `event.results` Objekt Werten kann nun auf die einzelnen Wörter in dem Array zugegriffen werden. Um eine Untersuchung auf einzelnen Worte einzuschränken wird über jedes Element in dem Array gearbeitet und mit einer Funktion zum Vergleichen zweier Strings auf ein Kommando geachtet. Diese Funktion `userSaid(str, string)` gibt einen boolschen Wert retour und ermöglicht somit das Auslösen eines Events auf dem Media Element. Da die Events für Media Element gleich sind, muss hier nicht auf den Unterschied zwischen Audio und Video Elementen geachtet werden.

Programm 5.11: Funktion für die Überprüfung der Sprachbefehle und Manipulation der Media Elemente.

```
/**
 * speechCommands(id) for checking all relevant commands for the media
 */
function speechCommands(id){
    // Define a threshold that the recognition results are worth looking at
    var confidenceThreshold = 0.5;

    // Simple function that checks existence of s in str
    var user = function(str, s) {
        return str.indexOf(s) > -1;
    }

    record.onresult = function(e) {
        media = id;

        if(e.results.length){
            for (var i = e.resultIndex; i < e.results.length; ++i) {
                var str = e.results[i][0].transcript;

                // Play the video
                if (user(str, 'start') || user(str, 'play')) {
                    document.getElementById(media).play();
                }
                // Stop the video
                else if (user(str, 'stop') || user(str, 'pause')) {
                    document.getElementById(media).pause();
                }
         ...
    };
}
```

Wurde ein Event ausgelöst, wird durch die `record.continuous` Einstellung weiter auf nachfolgende Kommandos gewartet. Auch nach Beenden des ausgelösten Events auf dem Media Element wird immer noch auf Kommandos geachtet. Erst bei der Navigation zu der nächsten Folie wird der Auftrag zur Aufnahme der Sprache unterbrochen. Wird jedoch wieder ein Media Element auf der nächsten Folie entdeckt beginnt die Aufnahme wieder und das erhaltene Array wird auf Kommandos untersucht.

Kapitel 6

Evaluierung

Basierend auf der Implementierung wurde ein Prototyp erstellt und durch Evaluierungsmethoden getestet. Durch die Evaluierung mittels Probanden soll auf Probleme bei der Verwendung des Protoypen hingewiesen werden. Bei den Probanden wurden Experten als auch mögliche Endbenutzer in Betracht bezogen. Für einen solide Ausführung der Evaluierung werden zuerst die Methode festgelegt und ein Ablauf der Durchführung mit Probanden festgelegt. Die daraus resultierenden Ergebnisse werden im Zuge der Evaluierung durch den Prototypen aufgelistet und lassen auf Problemen und Verbesserungen schließen.

In Anlehnung an die Masterarbeit[10] von Thomas Peintner wurde die Evaluierung ähnlich aufgebaut und durchgeführt.

6.1 Methoden

Zur Evaluierung des Prototypen werden empirische und analytische Methoden verwendet um die Gebrauchstauglichkeit mit potenziellen Endbenutzern zu überprüfen. Diese Verfahren werden definiert als „Usability Inspection“ und „Usability Test“ Methoden für Software oder Hardware.

Bei dieser Evaluierung sind die angewendeten Methoden sowie die Probanden bekannt[6]. Für diese Evaluierung wurden zwei „Usability Inspection“ Methoden ausgewählt, die Heuristische Evaluierung und das Aktion-Analyseverfahren, sowie eine „Usability Test“ Methode, die Think Aloud Methode. Wobei erstere zu den analytischen und zweitere zu den empirischen Methoden zählen.

6.1.1 Heuristische Evaluierung

Die Heuritische Evaluierung (HE) ist eine sehr informative Methode zur Erfassung von Evaluierungsdaten. Jakob Nielsen entwickelte gemeinsam mit Rolf Molich Heuristiken die zur Analyse von User Interface Design genutzt

werden. Diese definierten Heuristiken können in [9] nachgeschlagen werden. Hier werden Experten zur Evaluierung herangezogen, welche auf die Einhaltung der Heuristiken achten. Für die Evaluierung werden drei bis fünf Experten benötigt, wobei Personen teilnehmen sollen, die Erfahrung im Bereich Usability haben und eventuell im Fachgebiet des zu evaluierenden Systems. Es könnten auch unerfahrene Personen als Tester in Betracht bezogen werden, wobei diese Ergebnisse nicht optimal sind.

Die Grundlage für die Evaluierung liegt darin, dass ein Evaluierender das zu untersuchende Interface alleine testet. Es wird die Kommunikation zwischen den Evaluierenden somit unterbrochen damit die Ergebnisse nicht untereinander beeinflusst werden und unverfälscht sind. Während ein Evaluierender das Interface untersucht, wird dieser unterschiedliche interaktive Elemente erkennen und soll diese mit den Usability Prinzipen vergleichen, den Heuristiken. Nach abgeschlossener Evaluierung werden alle Ergebnisse gemeinsam mit den Teilnehmern diskutiert und entsprechend in eine Liste aufgenommen, welche schlussendlich sämtliche Probleme beinhalten soll, welche die Heuristiken nicht entsprechen.

6.1.2 Formale Analyse

Die Formale Analyse gehört zu dem Aktion-Analyseverfahren[6, S. 73] (Action Analysis Method), zu welchem auch die „back-of-the-envelop“ Aktionsanalyse[1] zählt. Bei beiden liegt der Schwerpunkt mehr auf den Aktionen des Probanden als auf seinen Aussagen. Für die Evaluierung wurde die Formale Analyse verwendet.

Die Formale Analyse erfordert eine genaue Untersuchung der Aktionen, die von einem Probanden während eines Tasks[2] leistet. Diese Methode wird auch „keystroke level“ Analyse genannt. Ein Task kann mehrere Aktionen enthalten bzw. wird in mehrere Aktionen aufgeteilt und bei jedem Proband wird die Dauer der Durchführung notiert. Für die Probanden werden verschiedene Aufgaben vorbereitet. Wobei der „Schwierigkeitsgrad“[3] steigend ist und somit die Lernfähigkeit des Probanden testet. Dadurch sollte sich auch die Dauer der Durchführung gleichmäßig verhalten, da Erlerntes schneller umgesetzt wird und somit mehr Zeit für Neues bleibt. Das Aktion-Analyseverfahren gibt Auskunft über das Verhalten der Benutzer und der Dauer der Ausführung von Aufgaben.

[1] Mit „back-of-the-envelop“ Aktionsanalyse ist eine überschlagsmäßige Aktionsanalyse gemeint. Dieses Verfahren ist weniger aufschlussreich für die Evaluierung jedoch schneller umzusetzten.

[2] Ein Task ist eine Aufgabenstellung an den Probanden, welche er mit dem zu evaluierendem System durchführen soll. Ein Beispiel wäre das Starten eine Videos mit einem gegebenem Sprachbefehl.

[3] Der Schwierigkeitsgrad wird durch Kombination von mehreren Schritten hintereinander definiert.

6.1.3 Think Aloud Methode

Die Think Aloud Methode (THA) ist eine aussagekräftige Methode zur Evaluierung der Usability[6, S. 73]. Bei dieser Methode werden Endbenutzer als Probanden zum Testen des Systems eingesetzt. Wie schon der Name der Methode aussagt, müssen die Probanden jeden Gedanken während der Benutzung des Systems aussprechen. Durch das Äußern der Gedanken eines Probanden kann nachvollzogen werden, wie ein Proband das System sieht bzw. versteht. Somit kann festgestellt werden ab welchem Zeitpunkt ein eventuelles Fehlverstehen oder -verhalten seitens des Benutzers entstehen könnte.

Die Probanden erhalten bestimmte Aufgaben, welche sie mit dem System lösen müssen. Mit Beginn der Aufgabe muss der Proband jeden Gedanken aussprechen, und er wird dabei beobachtet. Die Aussagen werden protokolliert und lassen später auf mögliche Probleme in bestimmten Situationen im Umgang mit dem System schließen.

6.2 Durchführung

Um eine aufschlussreiche Evaluierung zu erhalten wurden drei verschiedene Evaluierungsmethoden durchgeführt. Die Kombination der Ergebnisse der jeweiligen Methoden bringt den Vorteil, fehlende Erkenntnisse durch eine andere Methode zu ergänzen. Weiters werden auch unterschiedlich qualifizierte Probanden eingesetzt und somit der Erkenntnisbereich erweitert. Nachfolgend wird beschrieben, wie die unterschiedlichen Evaluierungen aufgebaut und durchgeführt wurden.

6.2.1 Heuristische Evaluierung

Diese Methode der Evaluierung lässt Experten als Probanden zu, welche auf die Einhaltung der Heuristiken achtet.

Probanden

Bei den Probanden sollte es sich um Experten handeln, die im Bereich Usability Erfahrung besitzen und eventuell auch Fachkenntnisse im Bereich des Systems haben. Für die Evaluierung des Prototypen wurden drei weibliche Studentinnen und ein männlicher Student im Alter zwischen 22 und 26 Jahren aus unterschiedlichen Studiengängen der *University of Applied Sciences Upper Austria* in Hagenberg eingesetzt. Die Studenten haben alle Erfahrung mit Usability Tests und einen Einblick in Usability, welches im Zuge des Studiums erworben wurde, sowie auch das Verständnis für das System des Prototypen.

Trotz der gegebenen Erfahrung im Bereich Usability werden die Studenten nicht als hunderprozentige Experten angesehen. Somit war auch die

Verwendung von Heuristiken zur Evaluierung den Studenten unbekannt.

Vorbereitung

Der Prototyp für das Steuern der Präsentation wurde als webbasierte Anwendung implementiert und ist somit plattformunabhängig und braucht auch nicht vorab bei den Probanden installiert werden. Die Probanden dürfen ihr eigenes Notebook mit Browser und Internetzugang für die Evaluierung verwenden. Für den Fall, dass der Proband kein Notbook mit integrierter oder separater externer Webcam sowie Mikrofon besitzt, wird ein Laptop für den Probanden zur Verfügung gestellt, der den notwendigen Kriterien entspricht. Um mit der Evaluierung direkt beginnen zu können, muss eine HTML5 Präsentation zur Verfügung gestellt werden, die alle Elemente enthält, die durch das System gesteuert werden können. Entsprechend der Fachkenntnisse der Probanden wurde eine Präsentation im Vorfeld erstellt und von den Probanden mit Inhalt sowie Media Daten versehen.

Für die Evaluierung wurden Heuristiken angelegt, die in Anlehnung an den Heuristiken[4] von Nielsen und Molich, erstellt wurden. Die Tabelle (Tab. A.1) mit den aufgelisteten Heuristiken wurde für die Probanden vorbereitet sowie eine weitere Evaluierungstabelle (Tab. A.2) für das Dokumentieren von Problemen bei den Usability Prinzipien.

Durchführung

Vor Beginn der Evaluierung durch den Probanden wird diesem in einer kurzen Erklärung beschreiben, wie der geplante Ablauf der Evaluierung sein wird. Anschließend wird der Prototyp auf dem Notebook gestartet und die Informationstabelle sowie die Evaluierungstabelle an den Probanden ausgeteilt. Um einen Fremdeinfluss auf den Probanden zu vermeiden, führen alle Probanden die Evaluierung einzeln und getrennt von anderen aus. Somit konnte die Evaluierung gestartet werden. Für die Evaluierung wurde dem Probanden eine Liste mit Aufgabenstellungen gegeben. Diese Liste enthielt unterschiedliche Aufgabenstellungen, die der Proband ohne externe Hilfe ausführen muss und für die parallel laufende Formale Analyse erstellt wurde. Nach Beenden der Aufgabenstellung konnte der Proband nach eigenem Ermessen den Prototypen testen und auf Wunsch die Präsentation erweitern. Problematisch während der Evaluierung mit den Heuristiken war die Unsicherheit bei der Zuweisung der Heuristiken.

Die Dauer der Evaluierung betrug zwischen 40 und 70 Minuten und wurde mit der abschließenden Diskussion der Ergebnisse am Ende des Tages in der Gruppe beendet. Dabei wurden die offensichtlichsten Probleme genannt und in einer Tabelle inklusive einer Wertung der Probleme aufgenommen.

[4] http://www.nngroup.com/articles/ten-usability-heuristics/

Somit wurden alle Probleme erfasst, ebenfalls Probleme, die nur einmal aufgetreten sind. Diese wurden entsprechend gewertet und gehen somit nicht verloren.

6.2.2 Formale Analyse

Als nächstes Evaluierungsverfahren wurde die Formale Analyse des Aktion-Analyseverfahrens angewendet. Diese Variante bietet eine Auflistung von Tasks an, die vom Probanden ausgeführt werden müssen. Dadurch wird aufgelistet, an welchen Aufgaben ein Benutzer scheitert oder Probleme hat.

Probanden

Bei den Probanden für dieses Verfahren handelt es sich um dieselben Studenten, die an der Heuritischen Evaluierung teilgenommen haben. Da bei der ersten Evaluierung den Probanden Aufgaben gestellt wurden, konnte das Verhalten der Probanden vom Beobachter analysiert und entsprechend dokumentiert werden. Der Proband wurde zuvor informiert, dass seine Schritte bei der Aufgabenlösung dokumentiert werden.

Vorbereitung

Der Prototyp für bereits für die erste Evaluierung vorbereitet und wird ebenfalls bei diesem Verfahren verwendet. Für die Durchführung wurde eine Aufgabenstellung (Tab. A.3) für die Probanden vorbereitet und parallel zu dieser Aufgabenstellung liegt eine Tabelle (Tab. A.4) vor, die das Verhalten des Proband dokumentiert.

Durchführung

Vor Beginn der Analyse wird dem Probanden die Aufgabenstellung übergeben und kurz erklärt, dass die Aktionen des Probanden sowie die Dauer der Ausführung dokumentiert werden. Wichtig war es, den Probanden zu erklären, dass die Dauer keine Auswirkung auf seine Fähigkeiten hat um keine Stresssituation zu erzeugen.

Mit Beginn einer Aufgabenstellung wurde die Dauer der Durchführung dokumentiert. Die einzelnen Schritte, die der Proband für die Aufgabe ausführte, wurden in der Tabelle festgehalten. Führte ein Proband einen Schritt durch, der nicht dem natürlichen Ablauf entsprach, wurde dieser entsprechend aufgeschrieben und markiert. Diese besonderen Schritte sind für das Userverhalten ausschlaggebend. Nach Beenden einer Aufgabenstellung wurde durch den Probanden in der Aufgabenstellung vermerkt, ob die Aufgabe erfüllt oder abgebrochen wurde.

6.2.3 Think Aloud Methode

Die letzte Evaluierung wurde mittels der Think Aloud Methode und möglichen Endbenutzern als Probanden durchgeführt. Dieses Verfahren fordert die Probanden ihre Gedanken während einer Aufgabe laut auszusprechen und werden protokolliert für die spätere Feststellung der Erkenntnisse.

Probanden

Für diese Evaluierung wurden vier Frauen und drei Männer im Alter zwischen 25 und 39 Jahren als Probanden eingesetzt. Bei der Auswahl der Probanden wurde darauf geachtet, dass der ausgeführte Beruf auf einen Endbenutzer schließen lässt. Da es sich um ein System handelt, das die Steuerung einer Präsentation bestimmt, wurden Personen aus den Berufssparten gewählt, in denen Vorträge gehalten werden, vor Kunden präsentiert wird sowie in denen Präsentationen als Unterrichtshilfsmittel gelten.

Vorbereitung

Das Notebook mit gestarteter Anwendung wurde in einem Raum für den Probanden zur Verfügung gestellt. Für die Anwendung des Prototypen wurde eine allgemeine HTML5 Präsentation mit Inhalt für die Probanden erstellt. Die Aufgaben wurden in einer Tabelle (Tab. A.5) aufgelistet, die der Proband während der Evaluierung durchführen soll. Die Aufgaben wurden so erstellt, dass mit jeder weiteren Aufgabe die Durchführung schwieriger bzw. intensiver wurde.

Durchführung

Im Vergleich zu den beiden vorherigen Verfahren wurde bei der Think Aloud Methode die gesamte Evaluierung alleine mit dem Probanden durchgeführt. Somit wurde für jeden Probanden ein Termin vereinbart und die Dauer entsprechend kalkuliert um eine Testsituation ohne Zeitdruck zu erzeugen.

Vor der Evaluierung wurde dem Probanden die Think Aloud Methode genau erklärt um die Wichtigkeit der Aussagen des Probanden zu verdeutlichen. Es wurde darauf hingewiesen, dass nicht nur die Aktionen der Ausführung beschrieben werden sollen, sondern auch die Gedanken des Probanden zu den jeweiligen Aktionen. Anschließend wurde dem Proband die Aufgabenliste gegeben und jede Aufgabe kurz besprochen. Dem Probanden wurde mitgeteilt, dass vor jeder Aufgabe auftretende Unklarheiten jederzeit besprochen der könnten, da einige Probanden Bedenken äußerten bezüglich der korrekten Durchführung einer Aufgabe. Während der Durchführung einer Aufgabe wurden alle Bemerkungen sowie das Verhalten des Probanden dokumentiert.

Das Verhalten der Probanden war anfangs zögerlich, auf Grund der Angst einen Benutzerfehler auszuführen, welches sich aber nach und nach legte. Mit wachsender Sicherheit der Bedienung des Systems mussten die Probanden öfters gebeten werden, ihre Gedanken laut zu äußern. Die Dauer des Verfahrens lag zwischen 25 und 45 Minuten. Nach der letzten Aufgabe und einem abschließendn Gespräch mit dem Probanden wurde die Evaluierung beendet.

6.3 Ergebnisse

In diesem Abschnitt werden die festgestellten Probleme durch die Evaluierungsmethoden aufgelistet und mit möglichen Lösungsvorschlägen zur Verbesserung ergänzt.

6.3.1 Heuristische Evaluierung

Bei der Evaluierung wurden nach dem gemeinsamen Gespräche mit den Experten folgende Punkte als problematisch bezeichnet.

Position zum Gerät

Durch die Aufgabenstellung wurden die Probanden zum Verändern der Position und Wiederherstellen der ursprünglichen Position vor dem Gerät aufgefordert. Dadurch entstand mehrmals ein Konflikt mit der Erkennung von Handgesten und der Proband musste sich kurz neu positionieren.

Es wurde auch darauf hingewiesen, dass hier eine entsprechende Fehlermeldung bzw. Infomation für den Benutzer ausgegeben werden sollte. Mit einer einer Information zur Neupositionierung kann der Benutzer schneller auf das System reagieren und umgekehrt. Da jedoch oft die Präsentationfläche am Bildschirm des Gerätes gleich der Übertragungsfläche an Wänden oder Projektorflächen ist, wurde vorgeschlagen, diese Information als reine grafische Information auf der Präsentation darzustellen. Als grafische Information wurde ein kurzes Aufblitzen eines helleren Hintergrundes vorgeschlagen.

Gestenerkennung Information

Ein weiteres Problem stellte die zu schnelle Wiederholung einer Handgeste nacheinander dar. Wurde durch einen Probanden eine Geste zweimal hintereinander folgenden in einem kurzen Abstand durchgeführt, so wurde die zweite Geste ignoriert, da noch die erste ausgeführt wurde.

Als Lösung wurde eine weitere grafische Information vorgeschlagen, die sich auf die Transparenz der Präsentationfolie auswirkt. Bei leichter Transparenz soll für den Benutzer ersichtlich sein, dass eine Geste erkannt wurde

und ausgeführt wird. Ist die Folie nicht transparent kann der Benutzer eine weitere Geste ausführen.

Variierende Sprachbefehle

Bei den Sprachbefehlen für die Steuerung der Media Elementen gab es ebenfalls eine Anmerkung zum Thema Umfang der Sprachbefehle. Die aktuell eingebunden Sprachbefehle umfassen den groben Wortschatz für einen Benutzer. Jedoch sollte noch mehr auch auf die übliche Satzstellung von Benutzern geachtet werden.

Ein Lösungsweg der Experten war eine optinale Einbindung von Befehlen, die von den Benutzern definiert werden können. Diese Liste sollte auf Befehl und gewünschtes Wort eingeschränkt werden. Somit würde für jeden Befehl ein explizites Wort stehen.

Ein anderer Vorschlag war die Definition einer fixen Liste mit Befehlen, da auch bei anderen Systemen mit Sprachbefehl-Erkennung dem Benutzer Einschränkungen aufergelegt werden.

Navigation zur Überblick Ansicht

Mit der Option in die Ansicht „Überblick“ der Präsentation zu wechseln stellt sich ein weiteres Problem dar. Bei der Evaluierung wurde zwar in den Überblick gewechselt, jedoch wurde die Rückkehr für den Probanden schwierig. Ursprünglich wurde eine Handgeste für den Wechsel *in den Überblick* und eine Handgeste für den Wechsel *aus dem Überblick* erstellt.

Als beste Lösung wurde eine Handgeste vorgeschlagen um in und aus dem Überblick zu wechseln. Somit sollte die intuitive Verwendung von Handgesten wieder bestehen. Weiters wurde auch eine Information angefordert, die am Bildschirm in einer Infobox erscheint, um den Benutzer darauf hinzuweisen, dass er sich in der Übersicht befindet.

Einbindung Media Elemente

Mit den Experten wurde auch die Struktur der Präsentation durchgegangen und eine Vereinfachung im Bezug auf Media Elemente vorgeschlagen. Die Media Elemente werden mittels *id's* behandelt und vom System durch diese manipuliert. Grundsätzlich sollten die *id's* vom Ersteller der Präsentation hinzugefügt werden. Sollte jedoch auf eine *id* vergessen werden, kann das System das Media Element zwar erkennen jedoch nicht manipulieren.

Um ein Fehlen einer *id* zu vermeiden soll über das System den Media Elementen eine *id* zugewiesen werden. Somit muss vor dem Aufruf der Präsentation der Inhalt durch das System geprüft werden und gegebenenfalls verändert werden.

6.3.2 Formale Analyse

Die Formale Analyse zeigte das Verhalten der Probanden einige Aufschlüsse, die auch durch die Heuristische Evaluierung aufgelistet wurden. Folgende Erkenntnisse konnten aus diesem Verfahren gezogen werden.

Media Element Einbindung

Bei der Einbindung von Video oder Audio Elementen in der Präsentation wurde die *id* für das entsprechende Element nicht beachtet, wodurch das Media Element nicht manipuliert werden kann. Dieser Fall trat zwar nur einmal bei einer Evaluierung auf, weißt jedoch eine potentielle Gefahr auf, dass Benutzer diese *id* unabsichtlich vergessen könnten. Der Proband erkennt in erster Linie nicht den Grund für das Fehlverhalten des Systems und kann somit auch keine weiteren Schlüsse daraus ziehen. Das System erkennt zwar ein Media Element auf der aktuellen Folie jedoch keinen Fehler, da das System durch die fehlende *id* keine Manipulation ausführen kann und somit auch keine Sprachbefehle abgleichen kann.

Für die Absicherung für diesen Fall muss von dem System bei der Erkennung eines Media Elements eine *id* diesem Element zugewiesen werden. Ohne diese *id* kann das angesprochene Element nicht mit den Sprachbefehlen gesteuert werden.

Abstand zwischen Körper und Gerät

Die Distanz zwischen dem Probanden und dem Notebook war bei einigen Aufgabenstellungen zu verändern. Durch diese Änderungen musste der Proband oft seine Position erneut vor dem Gerät verändern, da er etwas zu weit entfernt oder zu nahe am Gerät war. Ist die Distanz zum Gerät nicht entsprechend, kann eine Handgeste nicht erkannt werden und es erfolgt keine Navigation. Dadurch kam es zu schnellen Wiederholungen von Handgesten und der Proband wurde verwirrt. Erst nach einer Korrektur der Position konnte das System eine Handgeste erkennen und umsetzen.

Um den Probanden mitzuteilen, dass die Positionierung zum Gerät nicht ausreichend oder optimal ist, sollte eine Information für diesen zur Verfügung gestellt werden. Bei dieser Information muss hervorgehoben werden, dass es sich um eine ungeeignete Positionierung vor dem Gerät handelt und nicht um eine schlecht erkannte Handgeste.

Handgesten Erkennung

Auch bei einer optimalen Distanz zu dem Gerät konnten bei zwei Probanden fehlerhafte Handgesten festgestellt werden. Mit fehlerhaften Handgesten werden Gesten bezeichnet, die dem System zu wenig Bildbewegung enthielten und somit nicht als Geste erkannt wurden. Dies führte bei den Pro-

banden zu ebenfalls schnelleren Wiederholungen der bestimmten Handgeste aber jedoch zu keinem Resultat.

Für diesen Fall wurde vorgeschlagen, dass eine Information erscheinen soll, die darauf hinweist, dass die Geste fehlerhaft bzw zu schnell ausgeführt wurde und nochmal wiederholt werden soll. Durch die Information wird eine zu schnelle Wiederholung unterbrochen und das System sollte die nachfolgende Handgeste entsprechend erkennen.

Exakte Aussprache eines Sprachbefehls

Die Anwendung von Sprachbefehlen auf Media Elementen funktionierte grundsätzlich bei allen Probanden ohne Zwischenfälle. Jedoch gab es bei unterschiedlichen Wörtern ein Problem mit der Aussprache und Erkennung durch die Web Speech API. Da das System nur mit Endresultaten arbeitet, die aus der Spracherkennen gefiltert werden, konnten oft die Sprachbefehle nicht erkannt werden, da der Proband eventuell das Wort nicht entsprechend schön gesprochen hatte.

Zur Vermeidung dieser Ausfälle der Erkennung der Sprachbefehle werden auch Zwischenergebnisse mitverwendet. Diese erkennen Wörter schneller und lassen einen besseren Vergleich mit den Befehlen zu. Zwar werden mehrere Daten durch diese Zwischenergebnisse an das System zurückgegeben, jedoch greift das System auch bei etwas schlechter ausgesprochenen Wörtern.

Lichtverhältnisse im Raum

Bei der Veränderung der Lichtverhältnisse analysiert das System die Verhältnisse neu. Bei einer extremen Abdunklung eines Raumes und in Kombination eines dunkel gekleideten Probanden kann das System kaum noch Handgesten erkennen.

Es sollte immer eine gewisse Lichtquelle für den Probanden und dem System gegeben sein.

6.3.3 Think Aloud Methode

Bei der Think Aloud Methode wurden mögliche Endbenutzer als Probanden herangezogen und es wurden ähnliche Probleme wie bei der Formalen Analyse festgestellt. Die Erkenntnisse durch diese Evaluierung werden nachfolgende aufgeführt.

Abstand zwischen Körper und Gerät

Bei den Probanden dieser Evaluierung wurde ein ähnliches Verhalten festgestellt wie bei den Probanden bei der Formalen Analyse. Wurde die Distanz

zwischen Gerät und Proband verändert, war sich der Proband anfangs unsicher, ob das Gerät die Geste erkannte und veränderte die Distanz erneut. Nach einigen Versuchen konnte der Proband ein gewissens Gefühl für den Abstand entwickeln und diese dann entsprechend einschätzen.

Auch hier wurde eine Information für den Benutzer vorgeschlagen, welche darauf hinweist, dass die Position vor dem Gerät verändert werden sollte.

Handgesten Erkennung

Bei einer optimalen Position vor dem Gerät konnten die Handgesten in vertikaler und horziontaler Folienebene von den Probanden entsprechend durchgeführt werden. Wurden Handgesten zu schnell hintereinander durchgeführt, konnte der Proband erkennen, dass nur eine Handgeste analysiert wurde.

Die Information für eine ungenaue Handgeste wurde auch hier gewünscht um Benutzer darauf hinzuweisen.

Navigation in die Überblick Ansicht

Bei der Navigation in die Überblick Ansicht wurde ebenfalls von den Probanden dieselbe Handgeste wieder verwendet um in die normale Ansicht zu kommen, da es für den Proband logisch erschien dieselbe wieder zuverwenden um zurück zu kommen.

Diese Option zur Steuerung wurde als ein besserer Lösungsweg vorschlagen, als dem Probanden zu erklären, dass eine eigene Handgeste für die Rückkehr existiert.

Exakte Aussprache eines Sprachbefehls

Die exakte Aussprache von den Befehlen für die Steuerung von Media Elementen war auch bei dieser Evaluierung für die Probanden problematisch. Zwar konnten die Probanden nach wiederholter richtiger Aussprache des Befehls einen Erfolg verzeichnen, jedoch ist es für die Durchführung der Aktion nicht zweckmäßig.

Die Einbindung von Zwischenergebnissen erzielte deutlich bessere Ergebnisse mit den Probanden. Mit den Zwischenergebnissen der Sprachbefehle waren die Probanden erfolgreicher und könnten wie gewünscht die Media Elemente entsprechend manipulieren.

6.4 Fazit

Die verschiedenen Evaluierungen zeigten weitgehenden, dass die Steuerungen für die Probanden grundsätzlich intuitiv sind und verstanden wurden. Durch die Formale Analyse und die Think Aloud Methode konnte das Verhalten der Benutzer entsprechend beobachtet werden. Die Erkenntnisse lassen darauf schließen, dass die Probanden bei den grundlegenden Schritten

für die Navigation keine Probleme hatten, wenn die Distanz von Proband zu Gerät entsprechend festgelegt wurde. Bei den Sprachbefehlen konnte eine Verbesserung durch Zwischenergebnisse erreicht werden und optimierte die Reaktion des Systems auf die Befehle. Wurde der Proband in eine Situation gebracht, in der er mit der Steuerung feststeckte, versuchte der Probande durch vorherige Handgesten wieder in eine für ihn kontrollierte Situation zu kommen.

Die Probleme, die durch die Evaluierung festgestellt werden konnten, sind zum Teil auf die Einschränkung der verwendeten Hardware Komponenten zurückzuführen. Der andere Bereich der Probleme ist auf die Einschränkungen durch das System zurück zuführen. Der Bereich der Sprachbefehle wurde im Zuge der Evaluierung schon verbessert. Das Informationssystem für fehlerhafte Handgesten sowie nicht optimale Distanz zum Gerät müssen noch entsprechend für den Benutzer adapiert werden.

Die Kombination von Handgesten und Sprachbefehlen als Webanwendung für HTML5 Präsentation hat bei der Evaluierung bei den Probanden positive Reaktionen hervogerufen.

Kapitel 7

Zusammenfassung

Ziel dieser Arbeit war es, eine Möglichkeit für eine kombinierte Steuerung durch Handgesten und Sprachbefehlen intuitiv für den Endbenutzer zu implementieren. Die Steuerung sollte dem Benutzer intuitiv fallen und ohne große Dokumentationen für den Benutzer verwendbar sein. Durch die bestehenden Anwendungen im dem Bereich für Handgesten und Sprachbefehl Erkennung konnte ein Überblick erstellt werden, welche Technologien verwendet werden sollten für die Erstellung eines kombinierten Systems. Diese Arbeiten boten eine solide Grundlage für das Konzept, welches eine Kombination von Gesten und Sprache darstellt, das auf eine Online Applikation anzuwenden ist.

Dabei wurde vor allem auf eine Anwendung geachtet, in welcher alle HTML5 Elemente eingebunden werden könnten und später für die Manipulation geeignet sind, wie zum Beispiel eine HTML5 Präsentation. Durch die Verwendung und Manipulation von HTML5 Elementen, könnte eine Struktur vorgegeben werden, die der Benutzer erstellen kann und das System einfach erkennt. Für die nötigen Informationen für Handgesten und Sprachbefehle werden Schnittstellen für Video- und Audiostream verwendet. Veranlasst durch den Benutzer kann das System auf diese Informationen zugreifen und die eingehenden Streamdaten verarbeiten. Der Videostream wird für den Bereich der Handgesten, welche ständig vom System abgegriffen werden müssen, von dem Audiostream getrennt. Der Audiostream wird nur für die Erkennung von Sprachbefehlen benötigt und muss somit nicht dauerhaft analysiert werden.

Durch die Verwendung von Gesten und Sprache sollte der Benutzer intuitiv das System anwenden können ohne vorab Erklärungen zu erhalten. Durch die erhaltenen Video- und Audiostream werden dann die Elemente manipuliert. Bei der Analyse der Handgesten wurden verschiedene Bibliotheken als Unterstützung in Betracht gezogen. Die Einbindung von speziellen Bibliotheken, welche im Bereich der Bildverarbeitung vorhanden sind, war jedoch durch partieller Ineffizienz nicht möglich. Somit wurde eine Un-

tersuchung der Bewegung im Bild als Analysemethode verwendet. Bei den Sprachbefehlen wurden vordefinierte Wörter als Kommandos festgelegt. Diese Sprachbefehle können nur in bestimmten Bereichen angewendet werden, in denen sich auch Media Elemente befinden. Der implementierte Prototype wurde im Zuge einer Evaluierung durch verschiedene Verfahren auf Usability untersucht. Die verwendeten Methoden wurden nach möglichst aussagekräftigen Resultaten ausgewählt. Weiters wurde auch für Auswahl der Verfahren Wert auf eine gemischte Probandengruppe geachtet, welche Experten sowie mögliche Endbenutzer beinhalten sollte. Dies sollte bestmögliche Erkenntnisse über das verwendete System erschließen.

Durch die Evaluierung wurde festgestellt, dass die Handgesten und Sprachbefehle von den Probanden grundsätzlich intuitiv verwendet wurden, um die entsprechend gewünschten Ziele zu erreichen. Aufschlussreich war, dass die Probanden grundsätzlich Probleme bei den Handgesten hatten, wenn die Position zwischen Person und Gerät nicht optimal war. Somit konnte festgestellt werden, dass ein Hinweis auf die Positionierung vor dem Gerät für den Benutzer wichtig ist, um die richtige Verwendung zu gewährleisten. Auch im Bereich der unterschiedlichen Befehle für das Steuern von Media Elementen gab es eine interessante Entdeckung. Bei den Experten wurde eine benutzerdefinierte Konfiguration gewünscht und bei den möglichen Endbenutzern wurde eine Erweiterung der Liste der Befehle angemerkt oder eine strikte Definition für die Befehle. Im Bereich der Handgesten Analyse wurde ebenfalls durch die Experten angemerkt, dass eine genauere Analyse möglicherweise die Resultate verbessern würde, da es bei einer schneller Wiederholung einer Handgeste keine Reaktion durch das System gab. Durch die abschließenden Gespräche mit den Probanden wurde im Allgemeinen festgestellt, dass die Kombination von Handgesten und Sprachbefehlen für die webbasierte Anwendung ein interessanter Lösungsweg ist. Es wurden auch Bedenken eingebracht, die sich mit der Effizienz des System auseinander setzten. Für den Redefluss bzw. den Ablauf der Präsentation wurden aber bei den meisten Probanden positive Reaktion geäußert.

Weitere Schritte für den Prototypen würden die Einbindung von Lösungswege der entstanden Probleme durch die Evaluierung bedeuten. Durch die unterschiedlichen Evaluierungsverfahren sind verschiedene Ansätze von Problemen festgestellt worden. Für eine entsprechende Lösung muss hier ein allgemeiner Weg gefunden werden. Anschließend würde eine weitere Evaluierung durchgeführt, um die Lösungswege zu untersuchen. Mögliche Erweiterungen für das Konzept wären benutzerdefinierte Konfigurationen. Dies betrifft nicht nur den Bereich der Sprachbefehle, sondern auch die Konfigurationen von Handgesten. Abschließend kann festgestellt werden, dass für den möglichen Endbenutzer ein System durch Kombination von Handgesten und Sprachbefehlen intuitiv ist und einen positiven Einfluss auf den Ablauf auf die Maniupulation von HTML5 Element hat.

Anhang A

Evaluierung des Prototyps

A.1 Auflistung der Heuristiken

Tabelle A.1: Heuristiken für die Evaluierung des Prototypen in Anlehnung an die Heuristiken von Jakob Nielsen und Rolf Molich.

	Heuristik
1	*Sichtbarkeit von Benutzeraktionen*
	Bei einer Interaktion mit dem System soll der Benutzer eine sichtbare Veränderung wahrnehmen zur Bestätigung der Erkennung einer Aktion.
2	*Übereinstimmung zwischen System und realer Welt*
	Das System soll dieselbe „Sprache" wie der Benutzer sprechen, hinsichtlich der Satzstellung, Aufbau und der Umgangsweise des Systems. Eine Handgeste oder ein Sprachbefehl sollte in den natürlichen Umgang des Benutzers liegen.
3	*Benutzerkontrolle und -freiheit*
	Benutzer erzeugen unabsichtlich eine Situation in der sie sich nicht mehr weiter bewegen können. Diese Situationen sollten einen Notaustieg bzw. eine aussagekräftige Information für den Benutzer anbieten.
4	*Einheitliche Struktur und Standards*
	Der direkte Umgang mit dem System soll den Benutzer nicht verwirren. Für ein Endergebnis soll ein einfacher Weg für den Benutzer vorhanden sein. Es soll eine defnierte Abfolge geben, die den Benutzer zum Endergebnis führt.
5	*Fehlervorbeugung*

Fortsetzung auf nächster Seite

Tabelle A.1 – Fortsetzung von vorheriger Seite

	Heuristik
	Um Benutzerfehler vorzubeugen, kann in bestimmte Situationen durch das System eingegriffen werden. Wichtig ist hier, dass der Benutzer erkennt, dass es sich um einen fehlerhaften Schritt handelt und eine Ausweichmöglichkeit nutzt.
6	*Erkennen statt erinnen*
	Die Steuerung des Systems soll dem Benutzer leicht fallen. Für die Verwendung soll der Benutzer die Aktionen intuitiv abwenden und nicht aus dem Gedächtnis abrufen müssen.
7	*Hilfestellung um aus Benutzerfehler zu lernen*
	Wird durch den Benutzer ein Fehler erzeugt, soll einen Informationen dem Benutzer zur Verfügung gestellt werden, die eine mögliche Lösung vorschlägt.

A.2 Evaluierungstabelle für Heuristiken

Tabelle A.2: Evaluierungstabelle für Probanden der heuristischen Evaluierung, H. Nr. = Nummer der Heuristik, Stufe (1 = leicht, 5 = schwer).

H. Nr.	**Problem**	**Stufe (1 - 5)**	**Lösungsvorschlag**

A.3 Aufgabenstellung für Formale Analyse

Für die Formale Analyse würden Aufgabenstellungen definiert und von den Probanden ausgeführt. Dafür würde eine HTML5 Präsentation mit bestehendem Inhalt dem Probanden zur Verfügung gestellt. Der Inhalt der Präsentation musste folgende Bestandteile beinhalten: Text, Grafiken, Videosowie Audiomaterial und vertikale Folienebenen für erweiterte Navigation.

Tabelle A.3: Tasks bzw. Aufgabenstellungen für die Probanden bei der Formalen Analyse.

Nr.	**Aufgabe**	**Erledigt**
1	Den Inhalt der Präsentation so verändern, dass der Ablauf der Präsentation nach eigenem Ermessen stimmt. Inhalte stehen zur Verfügung und sind unter folgendem Verzeichnis erreichbar: *slides/material/...* Video und Audio muss in der Präsentation enthalten sein.	
2	Abstand zwischen Körper und Gerät aufbauen und mit der Präsentation starten. Navigation der Präsentation nur in horizontaler Folienebene mit varrierendem Abstand zum Gerät.	
3	Navigation der Präsentation in vertikaler und horizontaler Ebene mit varrierendem Abstand. Unterschiedlich lange Pausen einlegen für schnelleres Wechseln einer Folie.	
4	Navigation zu einer Folie, auf der sich ein Video Element befindet. Starten des Videos. Bei der Hälfte des Videos eine Pause einlegen und nach einer kurzen Pause das Video weiterspielen lassen.	
5	Präsentation abhalten wobei der Standort vor dem Gerät verändert werden muss um eine reale Präsentation zu simulieren. Bei der Präsentation dürfen nur die Gesten und Kommandos verwendet werden.	
6	Navigation der Präsentation durchführen, anschließend die Lichtverhältnisse im Raum verändern und erneut die Präsentation navigieren.	

Tabelle A.4: Erfassung der Durchführung der Aufgabenstellung durch die Probanden bei der Formalen Analyse. A. Nr. = Nummer der Aufgabe, Status (Y = durchgeführt, N = abgeborchen)

A. Nr.	**Beobachtung**	**Zeit**	**Status (Y/N)**
1			
2			
3			
4			
5			
6			

A.4 Aufgabenstellung der Think Aloud Methode

Für die Think Aloud Methode wird den Probanden eine HTML5 Präsentation mit den vorgeschriebenem Inhalt zur Verfügung gestellt. Anhand dieser Präsentation richten sich die auszuführenden Aufgaben für den Probanden.

Folgende Tabelle wurde den Probanden gegeben für die Durchführung der Evaluierung:

Tabelle A.5: Aufgabenstellungen für die Probanden für die Think Aloud Methode.

Nr.	**Aufgabe**	**Erledigt**
1	Die Präsentation mit der Applikation öffnen. Nötige Zustimmungen geben und Position für die Präsentation einnehmen. Testen ob die Präsentation auf eine Handgeste reagiert.	
2	Die Präsentation einmal vollständig auf horizontaler Ebene durchgehen mit den Handgesten. Um den Ablauf einzuhalten muss der Inhalt jeder Folie vom Präsentierenden vorgetragen werden. Audio und Video Elemente werden ignoriert.	
3	Die Präsentation auf vertikaler und horizontaler Ebene durchgehen nach eigenem Befinden. Dabei wird die Position vor dem Gerät geändert und bei einer Geste zur Navigation die ursprüngliche Position wieder eingenommen.	
4	Navigation auf einen Folie, welche ein Audio Element enthält. Die Audio Datei starten und nach ungefähr der Hälfte der Zeit pausieren und dann wieder abspielen.	
5	Navigation auf eine Folie mit einem Video Element. Diese dann starten und abbrechen. Auf nächste Folie navigieren und wieder zurück auf die vorherige Folie. Video erneut starten und nach Belieben stoppen.	
6	Einige Folien der Präsentation durchnavigieren und dann in den Überblick der gesamten Präsentation wechseln durch die entsprechende Handgeste.	
7	Präsentation nach Belieben navigieren mit Postionswechsel und bei Auftreten eines Audio oder Video Elements dieses starten.	

Anhang B

Inhalt der CD-ROM

Format: CD-ROM, Single Layer, ISO9660-Format

B.1 Masterarbeit

Pfad: /

Baumgartner_Elisabeth_2014.pdf Masterarbeit (Gesamtdokument)

B.2 Abbildungen

Pfad: /Abbildungen

AbfrageBrowser.png	Abfrage an den Benutzer durch den Browser
Ablaufdiagramm.jpg	Ablaufdiagramm des Kontroll-Scripts
js-handtracking.png	js-handtracking Anwendungsbeispiel
KinectJoints.png	Kinect Joints der Microsoft Kinect Sensor
Konzeptablauf.png	Konzeptablauf
MediaStream.png	MediaStream Video und Audio Verarbeitung
Revealjs.png	Revealjs Webcam Projekt
SupportAudio.png	Browsersupport des Audio Elementes
SupportVideo.png	Browsersupport des Video Elementes
SupportWebRTC.png	Browsersupport der WebRTC API
SupportWebSpeech.png	Browsersupport WebSpeech API
VideoVoiceProjekt.png	Video Voice Projekt
WebcamSwiper.png	Webcam Swiper Projekt
WebRTCDetail.png	WebRTC Architektur Details
WebRTCArchitektur.png	WebRTC Architektur
WebRTCNativeAPI.png	WebRTC Native API Darstellung

x-webkit-speech.png . . x-webkit-speech Element Verwendung

B.3 Protoyp

B.3.1 Code

Die gesamte Implementierung des Prototypen befindet sich unter /Protoyp/-Code auf der CD-ROM.

B.3.2 Beispiel HTML5 Präsentation

Pfad: /Protoyp/Präsentation

presentation.html . . . . Beispiel HTML5 Präsentation für den Prototypen

B.3.3 Dokumentation

Pfad: /Protoyp/Dokumentation

ProjectReport.pdf . . . Projekt Report für den Prototypen

TechnicalDocumentation.pdf Technische Dokumentation

UserDokumentation.pdf Dokumentation für Anwendung des Prototypen für Benutzer

Quellenverzeichnis

Literatur

[1] Daniel C Burnett Alan B. Johnston. *WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web*. 2. Aufl. Digital Codex LLC (siehe S. 42).

[2] Moniruzzaman Bhuiyan und Rich Picking. „Gesture-controlled user interfaces, what have we done and what's next?“ In: *... of the Fifth Collaborative Research Symposium ...* (2009). URL: http://www.glyndwr.ac.uk/Computing/Research/pubs/SEIN_BP.pdf (siehe S. 15, 34).

[3] Günter Born. *HTML5. Referenz und Nachschlagewerk*. Markt + Technik Verlag, 2012 (siehe S. 15).

[4] Daniel C Burnett, Adam Bergkvist und Cullen Jennings. *Media Capture and Streams*. 2013. URL: http://dev.w3.org/2011/webrtc/editor/getusermedia.html (siehe S. 41).

[5] Ilya Grigorik. *High Performance Browser Networking*. O'Reilly Media, 2013 (siehe S. 25, 27, 42).

[6] Andreas Holzinger. „Usability engineering methods for software developers“. In: *Communications of the ACM* 48.1 (Jan. 2005), S. 71–74. URL: http://portal.acm.org/citation.cfm?doid=1039539.1039541 (siehe S. 55–57).

[7] Bruce Lawson und Remy Sharp. *HTML5*. Addison-Wesley Verlag, 2012 (siehe S. 16).

[8] Juan Mellado. *js-handtracking JavaScript Hand Tracking*. 2012. URL: https://code.google.com/p/js-handtracking/ (siehe S. 8).

[9] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann, 1993 (siehe S. 56).

[10] Thomas Peintner. „Entwicklung eines grafischen Query Designers für relationale Datenbanken“. Masterarbeit. Fachhochschule Hagenberg, 2013 (siehe S. 55).

[11] Glen Shires. *Voice Driven Web Apps: Introduction to the Web Speech API*. 2013. URL: http://updates.html5rocks.com/2013/01/Voice-Driven-Web-Apps-Introduction-to-the-Web-Speech-API (siehe S. 9).