Mobile Device Usage in Interactive, Co-located Presentations

Iris M. Schaffer

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juni 2016

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, June 27, 2016

Iris M. Schaffer

Contents

Vorwort

Dies ist **Version 2015/09/19** der LaTeX-Dokumentenvorlage für verschiedene Abschlussarbeiten an der Fakultät für Informatik, Kommunikation und Medien der FH Oberösterreich in Hagenberg, die mittlerweile auch an anderen Hochschulen im In- und Ausland gerne verwendet wird.

Das Dokument entstand ursprünglich auf Anfragen von Studierenden, nachdem im Studienjahr 2000/01 erstmals ein offizieller LaTeX-Grundkurs im Studiengang Medientechnik und -design an der FH Hagenberg angeboten wurde. Eigentlich war die Idee, die bereits bestehende *Word*-Vorlage für Diplomarbeiten "'einfach"' in LaTeX zu übersetzen und dazu eventuell einige spezielle Ergänzungen einzubauen. Das erwies sich rasch als wenig zielführend, da LaTeX, vor allem was den Umgang mit Literatur und Grafiken anbelangt, doch eine wesentlich andere Arbeitsweise verlangt. Das Ergebnis ist – von Grund auf neu geschrieben und wesentlich umfangreicher als das vorherige Dokument – letztendlich eine Anleitung für das Schreiben mit LaTeX, ergänzt mit einigen speziellen (mittlerweile entfernten) Hinweisen für *Word*-Benutzer. Technische Details zur aktuellen Version finden sich in Anhang A.

Während dieses Dokument anfangs ausschließlich für die Erstellung von Diplomarbeiten gedacht war, sind nunmehr auch *Masterarbeiten*, *Bachelorarbeiten* und *Praktikumsberichte* abgedeckt, wobei die Unterschiede bewusst gering gehalten wurden.

Bei der Zusammenstellung dieser Vorlage wurde versucht, mit der Basisfunktionalität von LaTeX das Auslangen zu finden und – soweit möglich – auf zusätzliche Pakete zu verzichten. Das ist nur zum Teil gelungen; tatsächlich ist eine Reihe von ergänzenden "'Paketen"' notwendig, wobei jedoch nur auf gängige Erweiterungen zurückgegriffen wurde. Selbstverständlich gibt es darüber hinaus eine Vielzahl weiterer Pakete, die für weitere Verbesserungen und Finessen nützlich sein können. Damit kann sich aber jeder selbst beschäftigen, sobald das notwendige Selbstvertrauen und genügend Zeit zum Experimentieren vorhanden sind. Eine Vielzahl von Details und Tricks sind zwar in diesem Dokument nicht explizit angeführt, können aber im zugehörigen Quelltext jederzeit ausgeforscht werden.

Zahlreiche KollegInnen haben durch sorgfältiges Korrekturlesen und kon-

struktive Verbesserungsvorschläge wertvolle Unterstützung geliefert. Speziell bedanken möchte ich mich bei Heinz Dobler für die konsequente Verbesserung meines "‘Computer Slangs"’, bei Elisabeth Mitterbauer für das bewährte orthographische Auge und bei Wolfgang Hochleitner für die Tests unter Mac OS.

Die Verwendung dieser Vorlage ist jedermann freigestellt und an keinerlei Erwähnung gebunden. Allerdings – wer sie als Grundlage seiner eigenen Arbeit verwenden möchte, sollte nicht einfach ("‘ung’schaut"’) darauf los werken, sondern zumindest die wichtigsten Teile des Dokuments *lesen* und nach Möglichkeit auch beherzigen. Die Erfahrung zeigt, dass dies die Qualität der Ergebnisse deutlich zu steigern vermag.

Der Quelltext zu diesem Dokument sowie das zugehörige LaTeX-Paket sind in der jeweils aktuellen Version online verfügbar unter

https://sourceforge.net/projects/hgbthesis/.

Trotz großer Mühe enthält dieses Dokument zweifellos Fehler und Unzulänglichkeiten – Kommentare, Verbesserungsvorschläge und passende Ergänzungen sind daher stets willkommen, am einfachsten per E-Mail direkt an mich:

Dr. Wilhelm Burger, Department für Digitale Medien,
Fachhochschule Oberösterreich, Campus Hagenberg (Österreich)
wilhelm.burger@fh-hagenberg.at

Übrigens, hier im Vorwort (das bei Diplom- und Masterarbeiten üblich, bei Bachelorarbeiten aber entbehrlich ist) kann kurz auf die Entstehung des Dokuments eingegangen werden. Hier ist auch der Platz für allfällige Danksagungen (z. B. an den Betreuer, den Begutachter, die Familie, den Hund, ...), Widmungen und philosophische Anmerkungen. Das sollte allerdings auch nicht übertrieben werden und sich auf einen Umfang von maximal zwei Seiten beschränken.

Kurzfassung

An dieser Stelle steht eine Zusammenfassung der Arbeit, Umfang max. 1 Seite. Im Unterschied zu anderen Kapiteln ist die Kurzfassung (und das Abstract) üblicherweise nicht in Abschnitte und Unterabschnitte gegliedert. Auch Fußnoten sind hier falsch am Platz.

Kurzfassungen werden übrigens häufig – zusammen mit Autor und Titel der Arbeit – in Literaturdatenbanken aufgenommen. Es ist daher darauf zu achten, dass die Information in der Kurzfassung für sich *allein* (d. h. ohne weitere Teile der Arbeit) zusammenhängend und abgeschlossen ist. Insbesondere werden an dieser Stelle (wie u. a. auch im *Titel* der Arbeit und im *Abstract*) normalerweise *keine Literaturverweise* verwendet! Falls unbedingt solche benötigt werden – etwa weil die Arbeit eine Weiterentwicklung einer bestimmten, früheren Arbeit darstellt –, dann sind *vollständige* Quellenangaben in der Kurzfassung selbst notwendig, z. B. [Zobel J.: *Writing for Computer Science – The Art of Effective Communication.* Springer-Verlag, Singapur, 1997].

Weiters sollte daran gedacht werden, dass bei der Aufnahme in Datenbanken Sonderzeichen oder etwa Aufzählungen mit "'Knödellisten"' in der Regel verloren gehen. Dasselbe gilt natürlich auch für das *Abstract*.

Inhaltlich sollte die Kurzfassung *keine* Auflistung der einzelnen Kapitel sein (dafür ist das Einleitungskapitel vorgesehen), sondern dem Leser einen kompakten, inhaltlichen Überblick über die gesamte Arbeit verschaffen. Der hier verwendete Aufbau ist daher zwangsläufig anders als der in der Einleitung.

Abstract

This should be a 1-page (maximum) summary of your work in English.

Im englischen Abstract sollte inhaltlich das Gleiche stehen wie in der deutschen Kurzfassung. Versuchen Sie daher, die Kurzfassung präzise umzusetzen, ohne aber dabei Wort für Wort zu übersetzen. Beachten Sie bei der Übersetzung, dass gewisse Redewendungen aus dem Deutschen im Englischen kein Pendant haben oder völlig anders formuliert werden müssen und dass die Satzstellung im Englischen sich (bekanntlich) vom Deutschen stark unterscheidet (mehr dazu in Abschn. **??**). Es empfiehlt sich übrigens – auch bei höchstem Vertrauen in die persönlichen Englischkenntnisse – eine kundige Person für das "‘proof reading"’ zu engagieren.

Die richtige Übersetzung für "‘Diplomarbeit"’ ist übrigens schlicht *thesis*, allenfalls "‘diploma thesis"’ oder "‘Master's thesis"’, auf keinen Fall aber "‘diploma work"’ oder gar "‘dissertation"’. Für "‘Bachelorarbeit"’ ist wohl "‘Bachelor thesis"’ die passende Übersetzung.

Übrigens sollte für diesen Abschnitt die *Spracheinstellung* in LaTeX von Deutsch auf Englisch umgeschaltet werden, um die richtige Form der Silbentrennung zu erhalten, die richtigen Anführungszeichen müssen allerdings selbst gesetzt werden (s. dazu die Abschnitte **??** und **??**).

Chapter 1

Implementation

1.1 Project Scope

As the aim of the present work is to explore ways of incorporating mobile devices into presentation workflows, the goal of the project was to use an easily extensible presentation library to then build the mechanisms discussed in the previous chapter **??**. As the focus was placed on the interaction possibilities between speaker and audience, the creation of the presentation for the speaker or the management of slides and presentations were out of the project scope. Therefore the server used for connecting different users to the presentation was kept as simple as possible, allowing any potential other developer to work with their own servers and technology stacks.

In total, a system with several ways of interacting with the presentation from mobile or desktop devices was created, putting emphasise on mobile-optimised views and navigation possibilities. This system includes synchronisation of navigation state and state changes between viewers and speaker, the possibility to add sub-slides during the presentation for the audience, a speaker-view showing the next slides and controls, real-time voting (both created on-the-fly and prepared beforehands) and the possibility to create different paths through the presentation. In the following the technologies used in the project will be analysed and described to then go into details on the implementation, problems and solutions of the mentioned components.

1.2 Technologies

The project generally tries to follow best-practices in web development and utilises modern CSS3 and JavaScript features and frameworks. The software is written in ECMAScript2015, makes use of the *node package manager*[1](short *npm*) for managing dependencies and *Babel* to transpile to

[1] https://www.npmjs.com/

ECMAScript5. Additonally to relying on CSS3 features, this project also uses *Sass*[2] as a CSS pre-processor. Media-queries allow for mobile-friendly views.

On the front end, which this project focuses on, the JavaScript library *React* is the framework of choice, additionally applying the *reactive programming* paradigm using *RxJS* to allow for a simpler interface for event-driven operations. The communication between client and server is handled by *socket.io*[3]. This section tries to introduce the reader to the main technologies used to establish a base on which the following technical implementation details can be understood.

1.2.1 ECMAScript2015 and Babel[4]

JavaScript undoubtly is an integral part of front end web development and since the emergence of server-side JavaScript with Node.js[5] and its package manager npm has developed into a programming language widely used by web developers ([**gpm-meta-transcompiler**]). Both PYPL[6] and TIOBE[7] programming language indices rank JavaScript among the top 10 programming languages (PYPL at 5, TIOBE at 7 at the time of writing) ([**gpm-meta-transcompiler**]). Stack Overflow's 2015 Developer Survey even places JavaScript as the number 1, most-used programming language with 54.4% and JavaScript, Node.js and AngularJS[8] all three rank amongst the top 5 languages developers expressed an interest in developing with ([**stackoverflow-developer-survey**]).

However, like any front end technology, JavaScript suffers from slow end user adoption, as a multitude of browser versions exist for different devices and operating systems and many people still do not auto-update their browsers. Another factor is the time it takes for browser-vendors to implement new ECMAScript standards (the standard behind JavaScript) and roll out said updates. This is exactly what is happening with the new ECMAScript standard, ECMA-262, commonly known as ECMAScript 2015 or ES6: Although the General Assembly has adopted the new standard in June 2015 ([**ecma2015**]), *Kangax' ECMAScript compatibility tables*[9] still show a fairly low level of adoption, especially among mobile browsers. ES6 makes JavaScript easier and more efficient to write by providing new semantics for default values, arrow-functions, template-literals, the spread operator or ob-

[2] http://sass-lang.com/
[3] http://socket.io/
[4] https://babeljs.io/
[5] https://nodejs.org/en/
[6] http://pypl.github.io/PYPL.html
[7] http://www.tiobe.com/tiobe_index
[8] https://angularjs.org/
[9] https://kangax.github.io/compat-table/es6/

ject destructuring ([**es6**]). It also makes JavaScript easier to understand and safer to develop, with the introduction of block-scoped variables (`let` and `const`) and finally offers native support of modules and promises ([**es6**]). As these features are all included in the new ECMAScript standard, it is safe to assume browser-vendors will implement them in the near future. Until then, developers who want to already make use of them, can *transpile* ECMAScript 2015 code into ECMAScript 5, which is exactly what Babel does. With over 650000 downloads in March 2015 (according to npm) and companies like Facebook, Netflix, Mozilla, Yahoo or PayPal using this transpiler ([**babel-users**]), Babel is the de facto standard solution to transpile to ECMAScript 5 and was also chosen for this project.

1.2.2 Reactive Programming

Another problem with JavaScript, although integral part of the reason for its high popularity, is its asynchronous nature. Especially when working with highly interactive parts, the prime example being user interfaces, sequential programming quickly gets too inflexible to handle complex, event-driven applications ([**reactive-programming-survey**]). But also on the server, the possibility to concurrently serve a multitude of different clients, is crucial. In these cases JavaScript offers *event listeners* – functions called once a certain event happens. However, these event listeners or *asynchronous callback* ([**reactive-programming-survey**]), oftentimes executes more asynchronous code and in turn has to wait for another event, and another one, and another one... which can result in something known and dreaded by most any JavaScript developer: *Callback Hell* (see programm 1.1).

Different approaches have been employed to lower the hurdle of writing asynchronous code, one of them being *promises*: A promise is a value, yet to be computed ([**reactive-vs-promises**]). A promise can be a) pending (if it has not been assigned a value yet), b) resolved (if it has been assigned a value) or c) rejected (if an error occurred). In ECMAScript 2015 promises these objects can then be queued using the `then` keyword, to execute asynchronous code in a certain sequence (see programm 1.2).

However, promises can still create nested callbacks, especially when chaining promises that rely on other promisesŕesolution ([**reactive-vs-promises**]). This is where *reactive programming* comes in: The reactive programming paradigm works with streams of events, in which every event is handled as a new value and all other parts depending on this value are re-computed upon arrival of such a new value. To demonstrate this I would like to use Bainomugisha et al.ś illustrative example of a simple addition [**reactive-programming-survey**]:

```
1 var v1 = 1
2 var v2 = 2
3 var v3 = var1 + var 2
```

Program 1.1: *Callback Hell* – Nested callbacks in JavaScript. Simplified method taken from a previous project, which authenticates a user, creates a new google calendar for them and then saves the user to oneś own database, to then redirect them. `{...}` is used to shorten the code, error-handling was also omitted in the example for simplicity.

```
router.get('/callback', function(req, res, next) {
  var code = req.query.code;
  var name = JSON.parse(req.query.state);

  // get token from oauth library
  oauth2Client.getToken(code, function (err, tokens) {
    // load configuration
    Configuration.findOne({}, function (err, configuration) {
      var calendar = google.calendar('v3');
      // save to google calendar
      calendar.calendars.insert({...}, function (err, cal) {
        var member = new Member({...});
        // save member to own database
        member.save(function(err, member) {
          return res.redirect(getRedirectionUrl(name) + '&success=true')
        });
      });
    });
  });
});
```

Program 1.2: *Promises* – Simple example of chaining ECMAScript 2015 promises with `then` and `catch`.

```
var promise = new Promise(function(resolve, reject) {
  asyncCall(function(error, data) {
    if (error) {
      reject(error); // reject the pending promise
    } else {
      resolve(data); // resolve the pending promise
    }
  })
})

promise
  .then(function(data) {
    // this is executed after asyncCall returns
    // other asyncronous calls can be placed here
  })
  .catch(function(error) {
    // this is executed if an error occurs somwehere along the way
  })
```

Program 1.3: *RxJS* – simplified example of the touch controls used to swipe to the next or previous slide. An Observable is created from the browser's `touchmove` event and is then transformed with `map` and `filter`, to in the end call the `navigate()` method with the direction the user swiped into.

```
1  this.moveObservable = Observable.fromEvent(document, 'touchmove')
2    // data: event object with array of touches
3    .filter(this.touchStarted) // only procede if touchstart is set to true
4    .map(this.toXY) // transform initial event data to latest touch's xy position
5    // data: {x: xPosition, y: yPosition}
6    .map(this.toDirection) // transform xy position to direction literal
7    // data: right/left/up/down
8    .do(this.resetTouchStart) // set this.touchStarted to false
9    // data: right/left/up/down
10   .subscribe(this.navigate); // call this.navigate() with direction data
```

In sequentially executed code, `v3` will hold the value 3, no matter if or how `v1` or `v2` change. In reactive programming, however, `v3` will be re-computed as soon as either one of the values it depends on change ([**reactive-programming-survey**]). This way for a drag-and-drop feature, for example, the move of the mouse, continuously sending its location, could directly alter the position of an element in a page. JavaScript does not directly support reactive programming, but other, more functional languages, which can be transpiled to JavaScript, do. Another way of adding reactive programming concepts to JavaScript is using a library, such as *Bacon.js*[10] or the one chosen for this project, *ReactiveX*[11]. ReactiveX provides libraries for a multitude of different programming languages, C, C++, Java and of course JavaScript among them. The latter, called *The Reactive Extensions for JavaScript* or short *RxJs*, allows for the simple creation of event streams (*Observables*) from browser events or promises directly and uses the same method names JavaScript developers are familiar with from array-methods, most notably and well-known `map`, to apply a method to every element in the incoming stream and `filter`, to only let a subset of events pass. These methods can be chained to sequentially alter a value (see programm 1.3).

Additionally to Observables, RxJs also knows *Subjects*, which combine both a source of events and a consumer of such. Subjects are Observables, but also Observers at the same time and can be used to broadcast values to several consumers ([**rxjs-docu**]).

[10] https://baconjs.github.io/
[11] http://reactivex.io/

1.2.3 React[12]

As this project concentrates on the front end, a mature JavaScript framework was searched for. After previous experience with the big and complex, but slow AngularJS, because of promising performance benchmarks ([**react-benchmarks**]) and simply to explore new JavaScript libraries, I decided to give React a try. Since Facebook started developing React in 2013, it has challenged existing approaches and set new standards in front end web development ([**introduction-to-react**]). Instead of creating an entire MVC framework for the front end, React really concentrates on the view by offering a way of creating independent, lightweight view components. This gives React the huge advantage of beating other front end frameworks by far in performance benchmarks ([**react-benchmarks**]). The arguably most important method these re-usable, lightweight components implement is the `render` method, defining what HTML or JSX[13] should be rendered by the browser:

```
1 export default class HelloWorld extends Component {
2   render() {
3     return (<h1>Hello World!</h1>);
4   }
5 }
```

The created Component can then be rendered into the virtual React DOM, JSX makes it possible to simply use the name of the component to create it:

```
1 ReactDOM.render(
2   <HelloWorld />,
3   document.getElementsByTagName('body')[0]
4 );
```

This would simply put an `h1` element with the text `Hello World!` into the `body` element of the HTML page. Additionally to the `render` method, components also have a *state* and *properties*(`props`), through which they can communicate with other components and maintain their inner state. `props` are passed in to the component during creation, in JSX this can be achieved by simply passing them in as XML attributes:

```
1   <HelloWorld greeting="Hi" />
```

This could in turn be used in the `HelloWorld` component's `render` method:

```
1   render() {
2     return (<h1>{this.props.greeting} World!</h1>);
3   }
```

The children of a component are also available through [this.props.children]. The `state` variable, on the other hand, is responsible for handling internal

[12] https://facebook.github.io/react/index.html
[13] https://facebook.github.io/jsx/

updates e.g. through user interactions ([**react-docu**]). To alter the state, the method `setState` can be used, which will cause an update and re-rendering of the component. So if in the above example, the word "World" should be changed to something else by the user, a text field with an event handler can be added inside the component:

```
1 // set default state and props...
2
3 componentWillMount() {
4   // create observable from change event on input
5   this.observable = Observable.fromEvent('change', this.refs.input)
6     .pluck('target', 'value') // extract input text
7     .subscribe(this.update) // call update with value
8 }
9
10 componentWillUnmount() { this.observable.unsubscribe() }
11
12 update(text) { this.setState({name: text}) }
13
14 render = () => (
15   <div>
16     <h1>{this.props.greeting} {this.state.name}!</h1>
17     <input value={this.state.name} ref="input" />
18   </div>
19 )
```

Now, whenever a user changes the text in the `input` field, the RxJS Observable will receive a new value (a change event). The `value` of the field is extracted on line 6 and used to then update the state in the `update` method, which is implicitly called with the value passed through the Observable chain.

The two methods `componentWillMount`, `componentWillUnmount` as well as `componentDidMount`, `shouldComponentUpdate`, `componentWillUpdate`, `componentDidUpdate` and `componentWillReceiveProps` are *lifecycle methods*, called whenever the component is created, updated or destroyed.

As an end note on React, and a transition to the core presentation library, I want to add that React components can be nested, which is at the core of the project developed for the present thesis. To make it as easy as possible for other developers to use the created libraries, a presentation is built just as a usual HTML page, using JSX to reference the custom React components (see program 1.4).

1.2.4 unveil.js[14]

As a presentation layer, this project uses the open-source JavaScript library *unveil.js*, which was developed by Leandro Ostera and myself in the beginning of the project and which I extended and adapted to my needs in an

[14] https://github.com/ostera/unveil.js

Program 1.4: Nested React components. In this example, a 1-slide-long presentation is created.

```
<UnveilApp>
  <Slide name="start">
    <h1>Unveil</h1>
    <h2>a meta presentation</h2>
    <Notes>Greet people and welcome everyone</Notes>
  </Slide>
</UnveilApp>
```

own fork[15] during the project. This fork will be covered in section 1.4 of this chapter, until then, a short overview of the different parts of unveil.js is given and key concepts of the library are discussed.

Generally, unveil.js operates in a 2-dimensional slide-space: Every slide can have a next and previous slide in x, as well as in y-direction. To generate the y axis, slides can be nested in other slides. These slides have an optional unique name as well as an index in the slide-tree, which they are identified by. As shown in program 1.4, slides are created inside the component `UnveilApp`, the core of unveil.js. This component configures and sets up the entire application based on optional configuration passed in as properties. There are a few concepts unveil.js is built around to allow for extensibility and configurability, namely *presenters*, *controls* and *modes*. Presenters define the way slides are rendered, e.g. show notes, upcoming slide or hide them. Controls control a part of the application, e.g. navigating from one slide to the next using the arrow keys on the keyboard. Modes are what allows a speaker to have different presenter and controls from a member from the audience. Each mode defines its own presenter and set of controls, the mode is determined by the url query parameter `mode`. This allows anybody using unveil.js to define new modes, presenters and controls and thereby extending the base library as they wish. A few of these are already defined, namely a default `Presenter`, `UIControls` to navigate using buttons, `KeyControls` to navigate using the keyboard and `TouchControls` to navigate with swipe-gestures on touch screens.

1.3 Project Structure

As the puropose of this project was not only to experiment with different ways of interacting with presentations using mobile devices, but also to create something worthwhile and contribute back to the vibrant open-source community, the project is separated into different repositories, which are all

[15] https://github.com/irisSchaffer/unveil.js

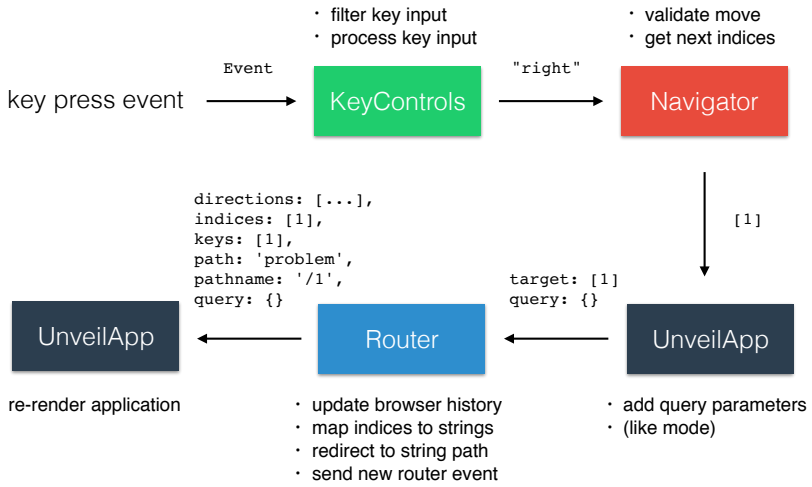


Figure 1.1: Navigation pipeline from user's key press to re-render of the presentation. The monospaced text next to the arrows symbolises the data transmitted. The `KeyControls` listen for key events and process them, to then send a navigation request to go *right* to the `Navigator`. This component then maps the direction to the next slide's indices (in this case 1). `UnveilApp` then adds other information necessary for the `Router`, which then is responsible for updating the browser history, mapping the indices back to a human readable url and sending out a new router event. In the end `UnveilApp` receives this event and re-renders the presentation.

available on GitHub. These can be installed using npm, therefore allowing developers to rely only on the parts they really need.

Network Synchronisation Layer: The first library of direct importance for the interaction between speaker and audience through personal devices is *unveil-network-sync*[16]. This rather small library relies on unveil.js and is responsible for connecting the client and the server through web sockets and enables the synchronisation of the current slide displayed between speaker, audience and projector. The implementation of the features will be discussed in detail in section 1.5.

Interactive Extension: As the name already suggests, this library is at the core of the present thesis: It sets up a dedicated view for the speaker, implements the insertion of additional slides and subslides and by that allows for sharing of content from the audience. The voting mechanism, as well as the creation of new votings on-the-fly, also live within this library. The

[16] https://github.com/irisSchaffer/unveil-network-sync

repository, called *unveil-interactive*[17] relies on unveil-network-sync for the socket-interaction. The library will be covered in section 1.6 of this chapter.

Server and Example Presentation: The last repository connected to this thesis is *unveil-client-server*[18], which includes a simple server as well as a real-world example of a presentation, which was used in the intermediate thesis project presentation as part of the Interactive Media course IM690, on the 2nd of February, 2016. In this chapter, a whole section was dedicated to the server (1.7), as well as to the example application (1.8), to separate concerns a bit more clearly and be able to conclude the chapter with a demonstration of how all parts discussed earlier play together in the final presentation.

1.4 Extended unveil.js

1.5 Network Synchronisation Library

1.6 Interactive Library

1.7 Server

1.8 Example Application

[17] https://github.com/irisSchaffer/unveil-interactive
[18] https://github.com/irisSchaffer/unveil-client-server

Appendix A

Technische Informationen

A.1 Aktuelle Dateiversionen

Datum	Datei
2015/09/19	`hgbthesis.cls`
2015/11/04	`hgb.sty`

A.2 Details zur aktuellen Version

Das ist eine völlig überarbeitete Version der DA/BA-Vorlage, die UTF-8 kodierten Dateien vorsieht und ausschließlich im PDF-Modus arbeitet. Der "'klassische"' DVI-PS-PDF-Modus wird somit nicht mehr unterstützt!

A.2.1 Allgemeine technische Voraussetzungen

Eine aktuelle LaTeX-Installation mit

- Texteditor für UTF-8 kodierte (Unicode) Dateien,
- `biber`-Programm (BibTeX-Ersatz, Version $\geq$ 1.5),
- `biblatex`-Paket (Version $\geq$ 2.5, 2013/01/10),
- Latin Modern Schriften (Paket `lmodern`).[1]

A.2.2 Verwendung unter Windows

Eine typische Installation unter Windows sieht folgendermaßen aus (s. auch Abschnitt **??**):

1. **MikTeX 2.9**[2] (zurzeit am einfachsten die 32-Bit Version, da nur diese

[1] http://www.ctan.org/pkg/lm, http://www.tug.dk/FontCatalogue/lmodern

[2] http://www.miktex.org/ – **Achtung:** Generell wird die **Komplettinstallation** von MikTeX ("'Complete MiKTeX"') empfohlen, da diese bereits alle notwendigen Zusatzpakete und Schriftdateien enthält! Bei der Installation ist darauf zu achten, dass die au-

das Programm `biber.exe` bereits enthält),

2. **TeXnicCenter 2.0**[3] (Editor-Umgebung, unterstützt UTF-8),
3. **SumatraPDF**[4] (PDF-Viewer),

Ein passendes TeXnicCenter-Profil für MikTeX, Biber und Sumatra ist in diesem Paket enhalten (Datei `_tc_output_profile_sumatra_utf8.tco`). Dieses sollte man zuerst über `Build` → `Define Output Profiles` in TeXnicCenter importieren. **Achtung**: Alle neu angelegten `.tex`-Dateien sollten in UTF-8 Kodierung gespeichert werden!

A.2.3 Verwendung unter Mac OS

Diese Version sollte insbesondere mit *MacTeX* problemlos laufen (s. auch Abschnitt **??**):

1. *MacTex* (2012 oder höher).
2. Die Zeichenkodierung des Editors sollte auf UTF-8 eingestellt sein.
3. Als Engine (vergleichbar mit den Ausgabeprofilen in TeXnicCenter) sollte *LaTeXMk* verwendet werden. Dieses Perl-Skript erkennt automatisch, wie viele Aufrufe von *pdfLaTeX* und *Biber* nötig sind. Die Ausgabeprofile *LaTeX* oder *pdfLaTeX* hingegen müssen mehrmals aufgerufen werden, zudem werden hierbei auch die Literaturdaten nicht verarbeitet. Dazu müsste extra die *Biber*-Engine aufgerufen werden, die jedoch noch nicht in allen Editoren vorhanden ist.

---

tomatische Installation erforderlicher Packages durch "'*Install missing packages on-the-fly: = Yes*"' ermöglicht wird (NICHT "'*Ask me first*"')! Außerdem ist zu empfehlen, unmittelbar nach der Installation von MikTeX mit dem Programm `MikTeX` → `Maintenance` → `Update` und `Package Manager` ein Update der installierten Pakete durchzuführen.

[3] http://www.texniccenter.org/

[4] http://blog.kowalczyk.info/software/sumatrapdf/

Appendix B

Inhalt der CD-ROM/DVD

Format: CD-ROM, Single Layer, ISO9660-Format[1]

B.1 PDF-Dateien

Pfad: /

_DaBa.pdf	Master- oder Bachelorarbeit mit Instruktionen (Gesamtdokument)
_PrBericht.pdf	Praktikumsbericht (verkürzte Version der Bachelorarbeit)

B.2 LaTeX-Dateien

Achtung: Die folgende Auflistung soll nur den Gebrauch dieser Vorlage erleichtern. Es ist bei einer Master- oder Bachelorarbeit i. Allg. *nicht* notwendig, die zugehörigen LaTeX-Dateien aufzulisten (wohl aber projektbezogene Dateien, Ergebnisse, Bilder, Kopien von Online-Literatur etc.)!

Pfad: /

_DaBa.tex	Master-/Bachelorarbeit (Hauptdokument)
_PrBericht.tex	Praktikumsbericht (verkürzte Version der Bachelorarbeit)
vorwort.tex	Vorwort
kurzfassung.tex	Kurzfassung
abstract.tex	Abstract
einleitung.tex	Kapitel 1

[1] Verwenden Sie möglichst ein Standardformat, bei DVDs natürlich eine entsprechende andere Spezifikation.

diplomschrift.tex	Kapitel 2
latex.tex	Kapitel 3
abbildungen.tex	Kapitel 4
formeln.tex	Kapitel 5
literatur.tex	Kapitel 6
drucken.tex	Kapitel 7
word.tex	Kapitel 8
schluss.tex	Kapitel 9
anhang_a.tex	Anhang A (Source Code)
anhang_b.tex	Anhang B (Inhalt CD-ROM)
anhang_c.tex	Anhang C (Liste der Änderungen)
anhang_d.tex	Anhang D (LaTeX-Quellcode)
messbox.tex	Messbox zur Druckkontrolle
literatur.bib	Literatur-Datenbank (BibTeX-File)

B.3 Style/Class-Dateien

Pfad: /

hgbthesis.cls	LaTeX Class-Datei für Master- und Bachelorarbeiten
hgbtermreport.cls	LaTeX Class-Datei für Semesterberichte
hgb.sty	LaTeX Style-Datei für alle Hagenberg-Dokumente

B.4 Sonstiges

Pfad: /images

*.ai	Original Adobe Illustrator-Dateien
*.fh11	Original Macromedia Freehand-Dateien
*.jpg, *.png	Original Rasterbilder

Appendix C

Chronologische Liste der Änderungen

2002/01/07 `\newfloat{program}` repariert (auch ohne Chapter). Dank an Werner Bailer!

2002/03/06 Copyright-Notice an internat. Standard angepasst. Dank an Karin Kosina!

2002/07/28 "'Studiengang"' → "'Diplomstudiengang"'

2003/08/24 Neues Macro: `\Messbox{breite}{hoehe}` – zur Kontrolle der Druckgröße ohne PS-Datei. Erweiterungen für Bakkalaureatsarbeiten

2005/04/09 Diverse Korrekturen: Captions von Tabellen nach oben gesetzt. `caption` auf neue Versionen adaptiert. `subfigure` wird nicht mehr verwendet

2006/01/20 Adaptiert zur Verwendung als Praktikumsbericht (2. Bakk.-Arbeit)

2006/03/24 Fehler in `\erklaerung` beseitigt (Dank an David Schwingenschlögl)

2006/04/06 Verwendung von T1-Fontencoding zur besseren Silbentrennung bei Umlauten etc.

2006/06/21 Neu: Bachelorstudiengang / Masterstudiengang. Literaturverweise auf Bakk-Arbeiten. `upquote.sty` eliminiert (Problem mit TS1-Kodierung). Verwende Komma (statt Punkt) als Trennzeichen in Dezimalzahlen.

2006/09/14 Anmerkungen zum Thema Plagiarismus.

2007/07/16 Ergänzungen für Code-Listings (listings) und Algorithmen (`algorithmicx`). BiBTeX-Datei aufgeräumt, Verwendung der Literaturformate verbessert. Komma (statt Punkt) als Trennzeichen in Dezimalzahlen wieder entfernt. Verwendung der `ae`-Fonts eliminiert (`cm-super` Fonts müssen installiert sein, ab MikTeX 2.5). Beispiel für

Ersetzung in EPS-Dateien mit `psfrag`.

2007/10/04 Version 5.90: Das Laden der Pakete `inputenc` (Option `latin`) und `graphicx` (Option `dvips`) aus der Hauptdatei in die `sty`-Datei übertragen; `upquote` funktioniert nun. Paket `eurosym` ergänzt für Euro-Symbol (Anregung von Andreas Doubrava). Problem mit `color`-package repariert (gerasterter PDF-Ausdruck). Hinweise bzgl. Literatur ergänzt (`month`, `edition`), BibTeX-Datei gesäubert. Hinweis zum Einfügen von vertikalem Abstand zwischen Absätzen. Mathematik aufgeräumt, Verwendung von `amsmath`, Fallunterscheidungen. Diverse Änderungen bei Tabellen und Programmkode. Beispiele für BibTeX-Angaben von Spezialquellen: Audio-CDs, Videos, Filme. Einbinden von Dateien mit `\include{..}` Neue Datei: `_SimpleReport.tex` für kurze Reports (Projekte etc.).

2007/11/11 Version 5.91: Hinweise zur Einstellung der Output-Profile in TexNicCenter, Inverse Search Einstellung in YAP im Anhang.

2008/04/01 Version 6.00beta – kompletter Umbau! Auslagerung der Dokumenten-relevanten Teile in eine eigene *class*-Datei (`hgbthesis.cls`) mit Optionen. Die neue Style-Datei `hgb.sty` ist nun unabhängig vom Dokumententyp und nicht mehr kompatibel mit älteren Versionen! Die Liste der Änderungen ist jetzt in der Datei `_ChangeLog.tex` (DIESE Datei) und diese wird im Anhang eingebunden. Heading-Style auf Sans Serif geändert (ohne grausliche "‘Caps"’).

2008/05/22 Neue Vorlage für Technical Reports (Klasse `hgbreport.cls`). Spracheinstellung nunmehr mit `babel`-Paket, Hauptsprache des Dokuments kann als Option der Klasse angegeben werden. Sprachumschaltung innerhalb des Dokuments funktioniert nun richtig. Mit der Sprachoption `german` wird automatisch die neue deutsche Orthographie (`ngerman`) verwendet. `babelbib` wird zur Formatierung des Literaturverzeichnisses verwendet (neue BibTeX-Style-Optionen!). Header werden nunmehr mit `fancyhdr`-Paket erzeugt. Versionsnummerierung von `.cls` und `.sty` Files wird beendet (ab jetzt gilt: *Datum = Version*).

2008/06/10 Neues Listing-Environment `PhpCode`; bei allen Listing-Evironments ist nun `mathescape=false` (kein Math-Mode nach `$`). Bug bei Sprachumschaltung auf `ngerman` beseitigt.

2008/08/15 Diverse Kleinigkeiten in Literaturangaben überarbeitet (Dank an Norbert Wenzel), Spracheinstellung vereinheitlicht, Umlaute in `.bib`-Datei ersetzt.

2008/10/15 Zusätzliche Hinweise zur MikTeX-Installation (Windows) sowie LaTeX unter Mac OS X und Linux. Liste der Abkürzungen ergänzt.

2008/11/15 Diverse Schreibfehler korrigiert (Dank an Silvia Fuchshuber).

Hinweis auf `sloppypar`-Umgebung.

2008/12/09 BibTeX-Tools: neuer Hinweis auf JabRef ergänzt, BibEdit entfernt (ist nicht mehr auffindbar).

2009/02/09 `hgb.sty`: Option "`spaces"' zu `url`-Package ergänzt (ermöglicht gezielten Zeilenumbruch in URLs). Im allgemeinen Setup für `listings`: `keepspaces=true`; Obsoletes Environment `sourcecode` deaktiviert. Escape-Mode für `LaTeXCode`-Umgebung geändert. `_DaBa.tex`: Hinweis auf die Verwendung von `\urldef` für die Angabe von URLs in Captions. `diplom` (statt `master`) als Standard-Dokumententyp in `_DaBa.tex` ("`Diplomarbeit"'). Neuer Abschnitt zum Umgang mit "Quellenangaben in Captions". `literatur.bib`: alle URLs (bisher in `note`-Einträgen) auf `url={..}` geändert.

2009/04/14 Hinweis zum Einfügen einfacher Anführungszeichen ergänzt.

2009/07/18 Literaturangaben korrigiert und ergänzt.

2009/11/27 Experimentelle Version: Massive Änderungen, Umstieg auf `pdflatex`.

2010/06/15 Erstes Release der neuen Version mit `pdflatex`.

2010/06/23 Konflikt zwischen `pdfsync`-Package und `array`-Package (wird relativ häufig benutzt) durch `\RequirePackage[novbox]{pdfsync}` behoben. Seitenunterkante durch `\flushbottom` fixiert, variablen Absatzzwischenraum reduziert.

2010/07/27 Sprache der Erklärungsseite auf "`german"' fixiert (auch wenn die Hauptsprache des Dokuments Englisch ist).

2010/12/03 Anmerkungen und Beispiele zum Zitieren von Gesetzestexten und Videos (Zeitangabe) ergänzt. Hinweis auf `\nolinkurl{..}` zur Angabe von Dateinamen.

2011/01/29 Einbau der Creative Commons Lizenz und entsprechender Hinweis in Abschnitt **??**. Neue Makros `\strictlicense`, `\cclicense` und `\license{...}`. BibTeX-Einträge für Audio-CDs und Filme korrigiert, Beispiel für Online-Video ergänzt.

2011/02/01 Neues Makro `\betreuerin{..}` zur Angabe einer (weiblichen) Betreuerin.

2011/06/26 Umstellung der gesamten Literaturverwaltung auf `biblatex` mit dem Ziel, getrennte Abschnitte für verschiedene Kategorien von Einträgen im Quellenverzeichnis zu ermöglichen. Die Wahl fiel auf `biblatex` (es gäbe andere Optionen), weil damit BibTeX weiterhin nur einmal aufgerufen werden muss (und nicht für mehrere Dateien). Damit verbunden sind allerdings massive Änderungen bei der Syntax der BibTeX-Felder und es gibt auch mehrere neue Felder. Aktuell sind 3 Kategorien von Quellen vorgesehen, entsprechende Änderungen in hgbthesis.cls. Der klassische BibTeX-Workflow wird aktuell nicht mehr unterstützt, die Möglichkeit einer künftigen Dok-Option ist aber

vorgesehen. Das Literatur-Kapitel ist komplett überarbeitet, die .bib-Datei wurde ausgemistet. Neu ist die Empfehlung zur Aufnahme von Bildquellen in das Quellenverzeichnis, womit lange URLs in Captions (dort sind keine Fußnoten möglich) nicht mehr notwendig sind. "'Persönliche Kommunikation"' als Literaturquelle entfernt (den Inhalt von Interviews sollte man direkt im Anhang wiedergeben). Das verwendete Bildmaterial wurde erneuert, aktuell werden nur mehr Public Domain Bilder verwendet. Das Kapitel "'Hinweise für Word-Benutzer"' wurde endgültig entfernt. `\flushbottom` wieder auf `\raggedbottom` geändert, um übermäßige Abstände zwischen Absätzen zu vermeiden.

2012/05/10 Hinweis auf die in Österreich bislang nicht zulässige Verwendung von "'Masterarbeit"' entfernt, `master` ist nunmehr die Default-Dokumentenoption. Anmerkungen zu lästigen `biblatex`-Warnungen ergänzt. Angaben für Windows-Programmpfade auf Win7 angepasst, MikTeX 2.9 als Minimalerfordernis.

Überflüssige Makros `\damonat` und `\dajahr` endgültig entfernt, statt `\abgabemonat` und `\abgabejahr` ist nun das neue Makro `\abgabedatum{yyyy}{mm}{dd}` vorgesehen (unter Verwendung von internen Zählern). Zur Formatierung von Datumsangaben wir das `datetime`-Paket verwendet.

Neue Fassung der eidesstattlichen Erklärung (inkl. englischer Version).

PDF-Suche auf `synctex` umgestellt (`pdfsync`-Paket ist veraltet und wird nun nicht mehr verwendet).

Die älteren Dateiversionen von `algorithmicx.sty` und `algpseudocode.sty` (bisher explizit beigefügt) wurden weggelassen.

Hinweis auf die *Latin Modern Roman* OTF-Schriften ergänzt.

2012/07/21 Quellenverzeichnis: sprachabhängige Einstellung der Überschriften eingerichtet. Titel des Quellenverzeichnisses auf "'Quellenverzeichnis"' (DE) bzw. "'References"' (EN) fixiert. Makro `\MakeBibliography` hat damit keinen erforderlichen Parameter mehr.

2012/09/17 Wegen Änderungen im `biblatex`-package (Version 1.7, 2011/11/13) die Verwendung von BibTeX als backend eingestellt (`backend=bibtex8`).

2012/10/13 Option `lowtilde` beim URL-package eingestellt (erzeugt ~ statt ~).

2012/12/01 In Abschnitt **??** zusätzliche Code-Umgebungen ergänzt: `JsCode`, `PhpCode`, `HtmlCode`, `CssCode`, `XmlCode`.

2012/12/08 Die Code-Umgebungen in Abschn. **??** ergänzt und zur Verwendung von optionalen Argumenten erweitert (Hinweise in Abschnitt **??** auf die Argumente `firstnumber=last` und `numbers=none`). Quellenverzeichnis: den Eintragstyp `@software` für Games empfohlen und im Verzeichnis der Kategorie *avmedia* zugeordnet (Tab. **??** ergänzt). Game-Beispiel (von Manuel Wieser) und zusätzliche Tabelle **??** zur

besseren Übersicht eingefügt.

2013/05/17 Wichtigste Änderung ist die vollständige Umstellung auf **UTF-8** unter Beibehaltung des `pdflatex`-Workflows. Damit sind zahlreiche weitere Modifikationen verbunden:
Alle Dateien (auch `.cls`, `.sty` und `.bib`) wurden auf UTF-8 konvertiert. Damit sollte es auch keine Probleme mehr mit Umlauten und Sonderzeichen unter MacOS geben.
Die verwendete Standard-Schriftfamilie ist nun "‘Latin Modern"’ (`lmodern`). Sie ersetzt die "‘CM-Super"’ Schriften, mit denen es immer wieder Installationsprobleme gab. Weiters wird jetzt das `cmap`-Paket zur besseren Such- und Kopierbarkeit von PDFs verwendet.
Das `listings`-Paket wurde durch `listingsutf8` ersetzt und für Umlaute im Quellcode adaptiert. Eventuell sind weitere Adaptierungen notwendig.
`biber` (min. Version 1.5!) wird nun anstatt `bibtex` (unterstützt keine UTF-8 Dateien) verwendet, zusammen mit `biblatex` (Version 2.5). Die Anweisung `\bibliography` wird (wieder) verwendet, allerdings nun in der Präambel, um die `.bib`-Datei im Fileverzeichnis anzuzeigen.
Das Makro `\C` (für die Menge der komplexen Zahlen $\mathbb{C}$) musste wegen Problemen in der T1-Kodierung ersetzt werden und heißt nun `\Cpx`. Die Makros `\R`, `\Z`, `\N`, `\Q` und `\Cpx` können nun auch außerhalb des Mathematik-Modus verwendet werden.
Der DVI-PS-PDF Workflow wird ab dieser Version überhaupt nicht mehr unterstützt. Damit ist auch das `psfrag`-Paket nicht mehr verwendbar. Entspechende Hinweise wurden aus dem Text entfernt.
`hyperref` wurde auf UTF-8 umgestellt. Die grässlichen Standard-Rahmen und Farben der automatischen `hyperref`-Links wurden entfernt bzw. durch dezentere Farben ersetzt. Dadurch wird auch die Screen-Version der PDFs wieder lesbar.
Im Quellenverzeichnis wurde versuchsweise die `backref`-Option aktiviert. Damit werden bei allen Einträgen auch die zugehörigen Zitierstellen angegeben (erscheint durchaus sinnvoll).
Die bisherigen Korrekturen zur `biblatex`-Formatierung wurden entfernt, alles arbeitet nun mit Standard-Einstellungen. Die ursächlichen Probleme in `biblatex` scheinen in der aktuellen Version behoben zu sein.
Das Output-Profil für TeXnicCenter wurde für den neuen Workflow mit `biber` adaptiert und liegt nun in `_tc_output_profile_sumatra_utf8.tco`.
Das Windows-Script `_clean.bat` wurde entfernt, da TeXnicCenter nun ein eigenes "‘Clean Project"’-Kommando aufweist (in "‘Build"’).
Allgemeine Einstellungen zu *headings* und *biblatex* wurden aus der Datei `hgbthesis.cls` entfernt und in `hgbheadings.sty` bzw.

`hgbbib.sty` verlagert. Diese können nun unabhängig verwendet werden (s. Beispiel in `_TermReport.tex`).
Die Klassen-Datei `hgbtermreport.cls` wurde eliminiert, das Dokument `_TermReport.tex` basiert nunmehr auf der generischen LaTeX-Klasse `report` und verwendet keine eigene `.cls` Datei mehr.

2014/11/05 Neu: Logo auf der Frontseite bei allen Dokumententypen. Dazu gibt es ein neues Kommando `\logofile{pic}`, wobei `pic` der Name eine PDF-Datei im Verzeichnis `images/` ist. Falls *kein* Logo erwünscht ist, kann man die Zeile einfach weglassen oder durch `\logofile{}` ersetzen.
`hyperref`-Einstellungen: Einfärbung der Links wieder entfernt (`colorlinks = false`), weil beim Druck nicht abschaltbar. Stattdessen einheitliche (dezente) Rahmen für alle Linkarten. Zahlreiche Tippfehler eliminiert (Dank an Daniel Karzel).
Wegen eines Bugs in `biblatex 1.9` wurden die expliziten Abteilungen (`\-`) in `literatur.bib` vorübergehend entfernt (mit entsprechenden Folgen im Ergebnis). Der Bug soll in `biblatex 2.0` (derzeit noch nicht verfügbar) behoben sein.
Package `color` auf `xcolor` geändert. In `hgb.sty` neues "'Convenience-Makro"' `\etc` ergänzt. Output-Profil für TeXnicCenter/SumatraPDF (Windows) repariert, forward/inverse Search funktioniert nun (Datei `_tc_output_profile_sumatra_utf8.tco`).

2015/04/28 Paket `subdepth` (zur verbesserten Platzierung von Sub- und Superscripts) in hgb.sty ergänzt.

2015/07/14 Hinweis und Abhilfe für die (nicht automatische) Silbentrennung in zusammengesetzten Wörtern. Neu in `hgbheadings.sty`: `\RequirePackage[raggedright]{titlesec}` verhindert Blocksatz in Section-Überschriften (sehr unschön bei längeren Überschriften). Neu (in Abschn. **??**): Beispiel für die Verwendung des `overpic`-Pakets zur Annotierung von importierten Grafiken (verwendet zudem das `pict2e`-Paket).

2015/08/03 Logo-Datei auf `logo.pdf` umbenannt.

2015/09/17 Anweisung `\RequirePackage[utf8]{inputenc}` in die Dokumentendateien (`_xxx.tex`) verschoben (auf Anregung von Markus Kohm: "'. . . für die Verwendung von lualatex oder xelatex ist die Anweisung in hgb.sty störend, da bei diesen beiden aufgrund der nativen utf8-Unterstützung `inputenc` keinesfalls verwendet werden darf"').

2015/09/19 `hgb.sty` aufgeräumt. Makros `\@savesymbol` und `\@restoresymbol` aus `hgb.sty` entfernt (wurden nicht mehr verwendet; ggfs. Paket `savesym` als Ersatz). Makro `\optbreaknh` (optional break with no hyphen) auf `\obnh` umbenannt. Teile von `hgb.sty` in neue Dateien `hgbabbrev.sty` (div. Abkürzungen)

und `hgblistings.sty` (Code-Listings) verschoben. Hintergrundtönung der Code-Listings heller (auf 5% Grau) eingestellt. Layout: `\textfraction` auf 0.1 (statt fehlhafterweise 0.01) eingestellt. `hgbbib.sty`: `\clearpage` am Beginn des Quellenverzeichnisses entfernt (für `article`-Template).

2015/09/19 Alle `.cls` und `.sty` Dateien sind jetzt ANSI-codiert (Header eingefügt), wie laut CTAN-Richtlinien vorgesehen. Umlautzeichen wurden durch Makros ersetzt. Nur `hgblistings.sty` ist weiterhin UTF-8 (wegen notwendiger literaler Umlaute). `\RequirePackage[utf8]{inputenc}` steht sonst nur mehr am Beginn der jeweiligen (`.tex`) Haupttextdatei.

2015/10/29 Verwendung von "'In:"' im Quellenverzeichnis vor `article`-Einträgen (Eigenart von biblatex) durch passendes Makro in `hgbbib.sty` unterbunden (Dank an S. Dreiseitl).

2015/11/04 Hinweise in Abschnitt **??** auf TeXstudio unter Windows, Mac OS und Linux. Release-Ausgabe.

Appendix D

LaTeX-Quellkode

Hauptdatei _DaBa.tex

```
%% File encoding: UTF-8
%% äöüÄÖÜß  <-- keine deutschen Umlaute hier? UTF-faehigen Editor verwenden!

\documentclass[master,english]{hgbthesis}
% Zulässige Class Options:
%   Typ der Arbeit: diplom, master (default), bachelor, praktikum
%   Hauptsprache: german (default), english
%%-------------------------------------------------------------

\RequirePackage[utf8]{inputenc} % remove when using lualatex oder xelatex!

\graphicspath{{images/}}    % name of directory containing the images
\logofile{logo} % name of logo-PDF in images/ (or use \logofile{} for no logo)
\bibliography{literature}   % name of the BibTeX (.bib) file


%%%-------------------------------------------------------------
\begin{document}
%%%-------------------------------------------------------------

% Einträge für ALLE Arbeiten: --------------------------------
\title{Mobile Device Usage in Interactive, Co-located Presentations}
\author{Iris M.\ Schaffer}
\studiengang{Interactive Media}
\studienort{Hagenberg}
\abgabedatum{2016}{06}{27} % {YYYY}{MM}{DD}

%%%-------------------------------------------------------------
\frontmatter
\maketitle
\tableofcontents
%%%-------------------------------------------------------------

\include{vorwort} % ggfs. weglassen
```

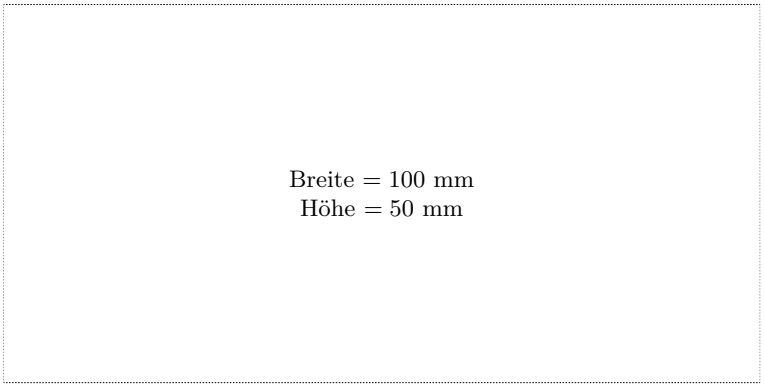
```
\include{kurzfassung}
\include{abstract}

%%%------------------------------------------------------------
\mainmatter         % Hauptteil (ab hier arab. Seitenzahlen)
%%%------------------------------------------------------------

\include{implementation}
%\include{einleitung}
%
%\include{diplomschrift}
%
%\include{latex}
%
%\include{abbildungen}
%
%\include{mathematik}
%
%\include{literatur}
%
%\include{drucken}
%
%\include{schluss}
%

%%%------------------------------------------------------------
%%%Anhang
\appendix
\include{anhang_a} % Technische Ergänzungen
\include{anhang_b} % Inhalt der CD-ROM/DVD
\include{anhang_c} % Chronologische Liste der Änderungen
\include{anhang_d} % Quelltext dieses Dokuments


%%%------------------------------------------------------------
\MakeBibliography
%%%------------------------------------------------------------

%%%Messbox zur Druckkontrolle
\include{messbox}

\end{document}
```

Anmerkung: Das sollte nur ein *Beispiel* für die Einbindung von Quellcode in einem Anhang sein. Der LaTeX-Quellkode der eigenen Abschlussarbeit ist meist *nicht* interessant genug, um ihn hier wiederzugeben!

References

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —