Entwicklung einer Applikation für die Darstellung von interaktiven Diagrammen im Web

Max Mathys

MNG Rämibühl

Betreuende Lehrperson: David Sichau

4. Januar 2016

Maturitätsarbeit

Abstract

In dieser Arbeit wurde untersucht, wie Punkt- und Liniendiagramme durch Interaktion verbessert werden können, sodass sich der Benutzer effizienter mit dem dargestellten Datensatz auseinandersetzen kann und die Analyse und das Verständnis der Daten erleichtert werden.

Eine Webapplikation für die Darstellung von 2- und 3-dimensionalen Datensätzen wurde entwickelt. Die Technologie einer Webapplikation wurde vorgestellt (JavaScript, HTML, CSS, SVG), das Applikationsdesign (Benutzeroberfläche, Konfiguration, Filtering, Mapping, Rendering) und Probleme, die während der Entwicklung auftraten, wurden dokumentiert. In dieser Arbeit wurden Konzepte der Datenvisualisierung (Visualisierungspipeline) und des Designs von Benutzeroberflächen (Information-Seeking Mantra) erläutert.

Bei den entwickelten interaktiven Diagrammen wurden verschiedene Interaktionsmethoden implementiert, die geläufig sind (wie z.B. Zoom, Filter, Details auf Abruf) oder in der Praxis nur selten verwendet werden (Reduktion von drei- auf zweidimensionale Punktdiagramme durch Projektion).

Inhaltsverzeichnis

Kapitel 1

Einleitung

Umgang mit grammatikalischen Geschlechtern: Aus Gründen der Spracheleganz wird vor allem die männliche Form (zum Beispiel „Benutzer“) verwendet. Die weibliche Form ist jedoch sinngemäss mitgemeint.

1.1 Bedeutung und Zweck von Diagrammen

Im Informationszeitalter sind Daten von grosser Bedeutung, der Datenfluss vergrössert sich ständig. Um Daten darstellen zu können, müssen sie zuerst gesammelt, sortiert und formatiert werden, bevor mit der Auswertung begonnen werden kann. Das Sammeln von Daten stellt oftmals keine besondere Schwierigkeit dar, das Auswerten, Darstellen und Interpretieren ist eine Herausforderung.

Die graphische Darstellung von Daten, das Diagramm, ist von grosser Bedeutung für die Gesellschaft. Man findet Diagramme überall: In etlichen Wissenschaften, wo sie nicht wegzudenken sind, in der Industrie, in Zeitungen, in Werbungen.

Das Ziel eines Diagramms ist die visuelle Repräsentation einer gegebenen Datenmenge, um damit eine effiziente Auswertung zu ermöglichen. Die Analyse, das Verständnis und die Kommunikation sollen erleichtert werden [1, Kapitel 1.1]. Der Leser sollte sich mit der Datenmenge auseinandersetzen können, ohne die Rohdaten selbst betrachten zu müssen. Je nachdem wie das Diagramm graphisch umgesetzt wird, können verschiedene Zusammenhänge und Informationen des Datensatzes hervorgehoben oder in den Hintergrund gestellt werden, was einen Einfluss auf die Vermittlung hat. Das Ziel eines Diagramms ist es jedoch, die Datenmenge möglichst unverfälscht darzustellen [1, Kapitel 1.1].

Konventionell werden Diagramme in Zeitungen, Artikeln abgedruckt. Der Autor trifft Entscheidungen, wie die Daten in graphischer Form dargestellt werden und erstellt auf dieser Basis ein Diagramm. Nach dem Druck kann es vom Leser betrachtet werden, jedoch kann dieser die Darstellung nicht mehr verändern, das Diagramm ist statisch. Er hat darum keinen Einfluss, wie die Daten in diesem *statischen Diagramm* dargestellt und an ihn vermittelt werden.

Durch das Aufkommen von modernen Computern und Smartphones haben sich die Möglichkeiten zur Darstellung erweitert: Dem Benutzer ist es möglich, mit dem Diagramm zu interagieren. Diese Maturaarbeit wurde vom Artikel *The Eyes Have It: A Task by Data Type for Information Visualizations* von B. Shneiderman [2] inspiriert. Der Artikel untersucht, wie ein Benutzer eine Programmoberfläche wahrnimmt und auf welche Weise er mit ihr interagiert. Auch beschreibt der Artikel, wie die Oberfläche aufgebaut sein sollte, damit der Benutzer sich im Programm orientieren kann, die Programmlogik versteht und das Programm ohne grossen Aufwand bedienen kann. Im Artikel stellt Shneiderman zudem das *Mantra der Informationssuche (Information-Seeking Mantra)* auf, das beschreibt, wie der Benutzer typischerweise mit einer Oberfläche interagiert: Überblick zuerst, Zoomen und Filtern, zuletzt Details auf Abruf [2].

Man findet unter anderem in Online-Zeitschriften solche *dynamischen Diagramme*, welche Daten darstellen, die für den Artikel relevant sind und interaktiv vom Benutzer bedient werden können. Die Abbildungen 1.1 und 1.2 demonstrieren, wie das Potential der Interaktion auf eine interessante Weise ausgenutzt werden kann.

In der Abbildung 1.1 ist ein Blasendiagramm dargestellt. Jede Blase stellt eine Firma dar, je nach Börsenwert ist die Blase unterschiedlich gross. Die Firmen sind in horizontaler Richtung nach ihrem Steuertarif geordnet, die verschiedenen Farben stellen die jeweilige Industrie, in der die Firma tätig ist, dar. Wenn die Maus über die Blase fährt, wird ein *Popup* sichtbar, das weitere Informationen zur Firma anzeigt wie der exakte Steuertarif, die absolute Menge an Steuern, die bezahlt wurde oder der Umsatz der Firma.

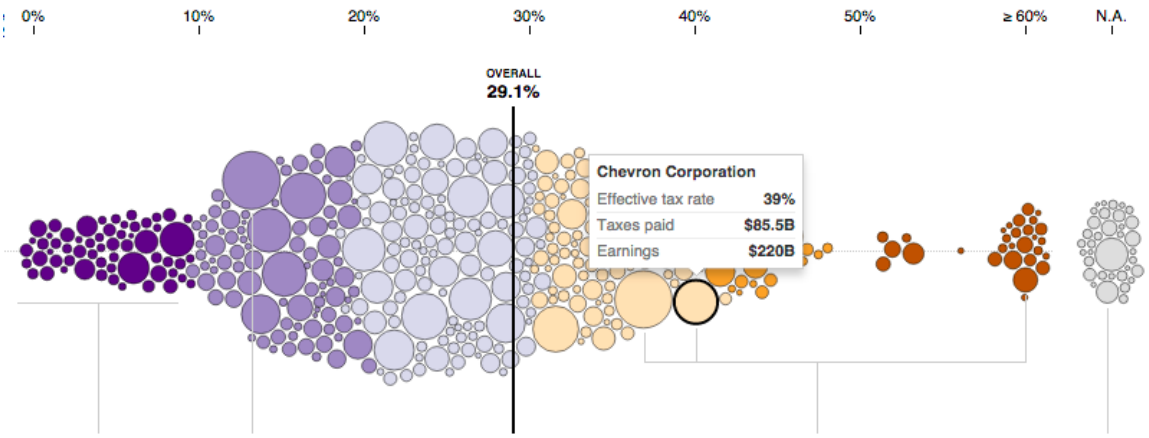

Abbildung 1.1: Ein Blasendiagramm, das Steuerabgaben und Steuersätze von US-Firmen darstellt. [3]

Die Webseite in Abbildung 1.2 berechnet anhand von vom Benutzer festgelegten Faktoren, ob der Kauf eines Hauses profitabler wäre als die Miete. Es sind Faktoren vorhanden wie der Zinssatz, die Entwicklung der Miete und der Wert des Hauses, der Steuersatz für Grundstücke, die Inflationsrate. Die verschiedenen Faktoren kann der Benutzer durch einen *Regler* anpassen. Dabei kann er durch die Höhe der Balken über dem Regler sehen, wie sich der Faktor auf das Resultat auswirkt (ob der Kauf oder die Miete profitabler wäre). Beim Verschieben des Reglers wird das Ergebnis automatisch aktualisiert, so sieht der Benutzer den Zusammenhang der Faktoren und der Berechnung.

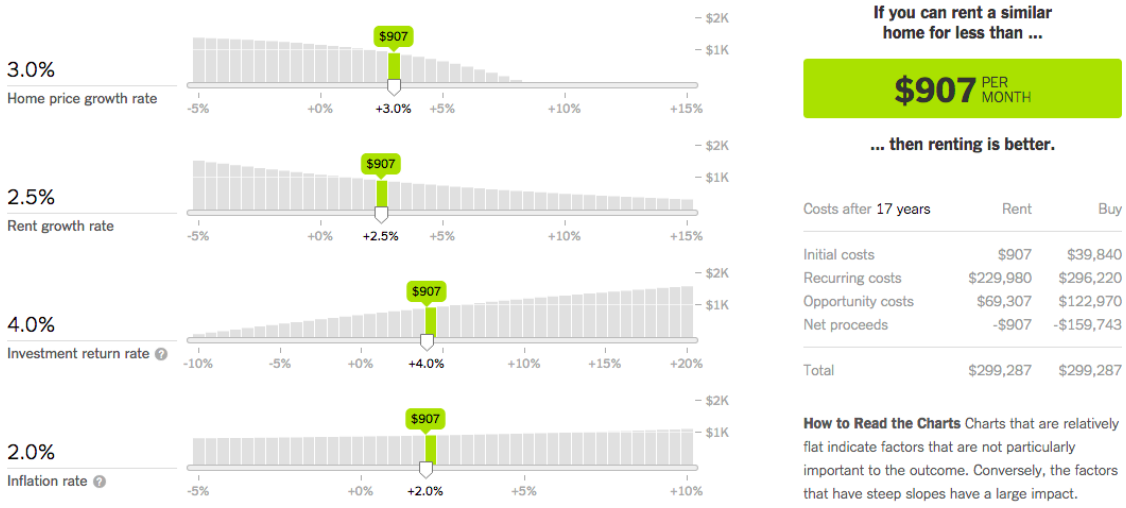

Abbildung 1.2: Interaktives Diagramm mit verstellbaren Reglern, das die Profitabilität des Kaufs eines Hauses darstellt. [4]

An den Diagrammen in Abbildung 1.1 und 1.2 kann man den Vorteil von dynamischen Diagrammen gut demonstrieren: In Abbildung 1.1 wird der dritte Teil des Mantras der Informationsvisualisierung angewendet, Details auf Abruf: Der Tooltip zeigt zusätzliche Informationen einer angewählten Firma an. Wegen Platzmangel wäre es nicht möglich gewesen, die gesamte Information aller Firmen in einem statischen Diagramm darzustellen.

In Abbildung 1.2 verändern sich die einzelnen Diagramme je nach Einstellung der Faktoren. Diese Verhaltensweise ist nicht umzusetzen in einem statischen Diagramm.

Die von Shneiderman beschriebenen Prinzipien für das Design von Benutzeroberflächen werden in dieser Arbeit berücksichtigt. Es werden interaktive, dynamische Diagramme entwickelt, die den Informationsertrag des Benutzers verbessern sollten. Er sollte sich mit den Daten effizienter auseinandersetzen können, besser verstehen und Spass an der Erkundung des Datensatzes haben [2, Kapitel 1].

1.2 Ziel dieser Arbeit

Das Ziel dieser Arbeit ist die Entwicklung einer eigenen Applikation, die dynamische Diagramme in einem Web-Browser darstellt. Sie soll den Prinzipien von Shneiderman folgen und dem Benutzer erlauben, eine bessere Analyse, ein besseres Verständnis des Datensatzes zu erlangen. Auch soll die Applikation dem Benutzer Spass machen und ihn ermuntern, sich mit dem Diagramm auseinanderzusetzen [2, Kapitel 1].

Das Design der Applikation wird auch aufgezeigt, mit dem diese Diagramme erstellt werden, welche Probleme bei der Entwicklung aufgetreten sind; verwendete Entwicklungswerkzeuge werden vorgestellt.

1.3 Wahl des Diagrammtyps für die Applikation

Es gibt sehr viele Arten von Diagrammen, welche sich jeweils für verschiedene Datenstrukturen eignen. Diese Applikation beschränkt sich auf zwei grundlegende Diagrammtypen: Das *Punktdiagramm* und das *Liniendiagramm*.

Das Punktdiagramm und das Liniendiagramm werden sowohl in der Wissenschaft als auch in den Medien oft verwendet. In diesen Diagrammen werden Skalen in der Ebene oder im Raum festgelegt, die ein Koordinatensystem beschreiben. Mindestens eine Achse dient als Skala der unabhängigen Variablen (der Dimension, die den Beobachtungsraum der Daten aufspannen). Die restlichen Achsen dienen als Skala der

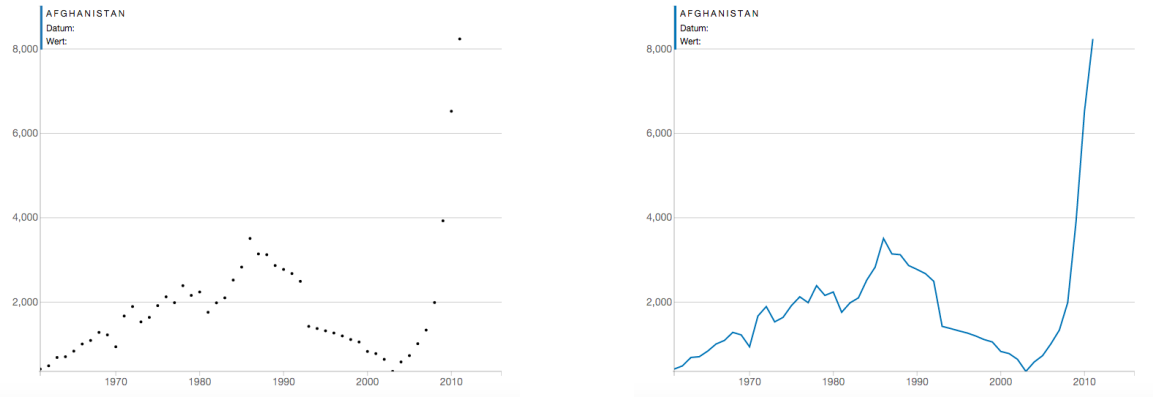

Abbildung 1.3: Beispiel von Diagrammen am Datensatz des CO_2-Ausstosses von Afghanistan. Links: Punktdiagramm. Rechts: Liniendiagramm mit linearer Interpolation.

abhängigen Variablen, Ausprägungen, die abgetragen werden und mit graphischen Symbolen, wie etwa Kreisen, Kreuzen oder Vierecken, markiert werden [1, Kapitel 5.2]. Beim Liniendiagramm werden zusätzlich die benachbarten Punkte durch eine Linie oder einen Kurvenabschnitt verbunden, was Trends und Strukturen der Daten deutlicher darstellt, und Datenpunkte, die durch Linien verbunden sind, werden zusätzlich gruppiert [1, Kapitel 5.2]. In der Abbildung 1.3 sind ein Punktdiagramm und ein Liniendiagramm dargestellt, die den Verlauf des CO_2-Ausstosses von Afghanistan zeigen.

1.4 Wahl der Programmiersprache für die Applikation

Interaktive Diagramme werden in der Praxis fast ausschliesslich für den Web-Browser entwickelt, wie auch in dieser Arbeit. Technologien wie HTML, CSS, SVG und JavaScript werden verwendet, sie ermöglichen die dynamische Manipulation durch den Benutzer. Diese Technologien werden durch praktisch alle neueren Smartphones, Tablets, Laptops und Desktops unterstützt.

Kapitel 2

Entwicklung der Applikation

Nachstehend wird beschrieben, wie die Applikation entwickelt wurde.

Der Code für diese Applikation kann im Git-Repository der Arbeit (https://github.com/mmathys/Matura-Paper) oder auf dem beigelegten Datenträger gefunden werden.

Anmerkung 20.02.2016 ETH Zürich. Es ist eine Demo-Seite zur Applikation vorhanden: Sie kann unter https://maturademo.github.io/ eingesehen werden.

2.1 Technologie

Für die Entwicklung einer Applikation für den Web-Browser werden verschiedene Technologien benötigt, die jeweils für einen einzelnen Aspekt der Applikation verantwortlich sind.

2.1.1 JavaScript

JavaScript ist die Programmiersprache, die der Web-Browser unterstützt. Die Programmlogik wird in dieser Sprache geschrieben.

Viele Programmierer tendieren dazu, den JavaScript Code in einem persönlichen Stil und auf nicht konsequente Weise zu formatieren, was die Lesbarkeit des Programmcodes, für den Autor als auch besonders für andere Betrachter, beeinträchtigt. Dies kann vor allem in umfangreichen Projekten, wo mehrere hundert Zeilen Code vorhanden sind, die Übersichtlichkeit markant verschlechtern, was etwa die Zusammenarbeit

mit anderen Programmierern beeinträchtigt und die Fehlersuche und -behebung (Debugging) erschwert. Da Open-Source von mehr und mehr Entwicklern praktiziert wird, gewinnen Syntaxkonventionen an Bedeutung. „Das Anwenden der [Syntax-] Konvention bedeutet, die Syntaxkonventionen der Community und den Belang der Lesbarkeit des Programmcodes über die persönliche Programmierweise zu stellen“ [5]. Für bessere Programmqualität und Lesbarkeit wird deshalb in dieser Arbeit der *JavaScript Standard Style* gebraucht, eine verbreitete Syntaxkonvention.

Die Applikation verwendet die Programmbibliothek *Data-Driven-Documents (D3)*. Sie wurde von Michael Bostock, Vadim Ogievetsky und Jeffrey Heer erstellt und dient zur Entwicklung von Visualisierungen im Web. Sie erleichtert die Benutzung des Document Object Model (DOM, siehe 2.1.3), ermöglicht effizienteres Debugging und verbessert die Leistung (*„Performance“*) der Applikation [6, Kapitel 1].

Für die Darstellung von dreidimensionalen Diagrammen wird die Bibliothek *three.js* [7] verwendet. Sie erleichtert den Zugriff auf die WebGL-Technologie des Browsers.

2.1.2 Buildsystem

Bei der Entwicklung einer Applikation ist die Verwendung von Buildsystemen unerlässlich. Es sind zahlreiche Open-Source Projekte vorhanden, die von vielen verschiedenen Personen entwickelt wurden, wie zum Beispiel *Gulp.js* [8], *Node.js (NPM)* [9]. Unter anderem wurden folgende Module für diese Buildsysteme verwendet: *Browserify* [10], *BrowserSync* [11], *uglify* [12] und viele weitere.

Im Endeffekt sind am Code des verwendeten Buildsystems mehrere Hundert Autoren beteiligt, da die Module wiederum selber auf weiteren Modulen aufgebaut sind.

2.1.3 Hypertext Markup Language (HTML), Document Object Model (DOM)

Hypertext Markup Language (HTML) ist in der Entwicklung einer Web-Applikation für den Inhalt (Text, Links, Bilder, Buttons...) zuständig. HTML verwendet die Extensible Markup Language (XML).

Das Document Object Model (DOM) ermöglicht die dynamische Manipulation der HTML-Elemente durch Schnittstellen in JavaScript-Programmen.

2.1.4 Scalable Vector Graphics (SVG)

Scalable Vector Graphics (SVG) ist ein Format für Vektorgrafiken. SVG-Bilder bestehen nicht wie andere Bildformate (JPG, PNG) aus Pixeln, sondern aus Elementen wie Kreisen, Ellipsen, Rechtecken oder Linien. SVG-Grafiken können im Browser dargestellt werden und durch JavaScript ebenfalls dynamisch manipuliert werden.

2.1.5 Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) beschreiben die Darstellung der anzuzeigenden Elemente, die in HTML-Dokumenten oder SVG-Grafiken vorkommen. In dieser Applikation wird das CSS-Framework *Basscss* [13] verwendet; es erleichtert die Gestaltung der Web-Oberfläche.

2.1.6 Originalität des Codes bei der Implementierung von Programmbibliotheken

Der Applikationscode und das Applikationsdesign werden selber erstellt, falls nicht anders angemerkt durch Kommentare im Programmcode.

Bei der Verwendung von Programmbibliotheken (D3, three.js, tween.js) und Buildsystemen (NPM, Gulp.js) wird der Code nach Anweisung der entsprechenden Dokumentation geschrieben.

2.2 Verarbeitung der Daten

Bevor das Diagramm im Browser dargestellt werden kann, müssen zuerst die Daten geladen und verarbeitet werden. Die Beispiele dieser Arbeit nutzen ausschliesslich Datensätze, die frei verfügbar sind [14]. Die Diagramme in Abbildung 1.3 benutzen einen Datensatz der Weltbank, der den CO_2-Ausstoss von Afghanistan beinhaltet.

Beim Prozess zur Veranschaulichung von Daten wird die *Visualisierungspipeline* durchlaufen [1, Kapitel 2.1]. Diese Pipeline stellt drei wesentliche Schritte des Prozesses dar: Die Datenaufbereitung (*Filtering*), die Erzeugung des Geometriemodells (*Mapping*) und die Bildgenerierung (*Rendering*).

2.2.1 Rohdaten

Comma-separated values (CSV)

Die Applikation verwendet als Datenformat *Comma-separated values* (*CSV*). Das CSV-Format wird verwendet, um Tabellendaten zwischen Programmen auszutauschen [15]. Tabellenzeilen werden in der CSV-Datei durch einen Umbruch dargestellt, Spaltenwerte durch Kommas. Optional steht in der ersten Zeile (Abbildung 2.1 links, Zeile 1) der Datei die Beschriftung der Spalten.

In Abbildung 2.1 wird die Funktion des CSV-Formats demonstriert.

```
1 Date,Value
2 2010-12-31,4220.717
3 2009-12-31,4352.729
4 2008-12-31,5555.505
5 2007-12-31,5067.794
```

Date	**Value**
2010-21-31	4220.717
2009-21-31	4352.729
2008-21-31	5555.505
2007-21-31	5067.794

Abbildung 2.1: Demonstration des CSV-Formats. Links: Die CSV-Datei in Rohtext. Rechts: Darstellung der Informationen der CSV-Datei in einer Tabelle.

Man verwendet das CSV-Format oft, weil es sehr einfach aufgebaut ist. Das Lesen von solchen Tabellenformaten ist ohne grossen Aufwand in Programmen umsetzbar.

Zudem beschränkt sich das CSV-Format nur auf die Vermittlung der Daten und beinhaltet keine Informationen zur Darstellung der Tabelle. Excel-Dateien (XLS/XLSX) hingegen speichern auch Daten zur Formatierung, auch Zeilengrösse, Textgrösse, Textformat und viele mehr. Für die Zwecke dieser Applikation sind diese Informationen irrelevant.

Wenn die Applikation die CSV-Datei laden und verarbeiten will, dann braucht sie zunächst zusätzliche Informationen zur Datei: Die *URL*, ein Zeichenstring, der eine über das Internet zugängliche Ressource repräsentiert [16].

Die Applikation lädt die Datei durch eine *Ajax*-Anfrage (*Asynchronous JavaScript and XML*) herunter, bevor sie sie weiter verarbeitet.

2.2.2 Filtering

Das Filtering ist der Prozess, der Rohdaten in aufbereitete Daten umwandelt. Die Aufgaben des Filtering sind zum Beispiel die Vervollständigung, Reduzierung oder Korrektur der Daten, sodass sie in den folgenden Schritten der Visualisierungspipeline verwendet werden können [1, Kapitel 2.1].

Konvertierung in JavaScript-Objekte

Als erster Schritt wird in der Applikation die CSV-Datei in *JavaScript-Objekte* umgewandelt. Das Laden und *Parsing* der Datei von CSV zu Objekten ist von D3 implementiert.

JavaScript-Objekte werden im *JavaScript Object Notation*-Format (*JSON*) dargestellt.

Im Abbildung 2.2 ist der Prozess der Umwandlung ersichtlich: Es wird ein *Array* von allen Zeilen in der Tabelle (ausgenommen der ersten Zeile, wo die Spalten beschriftet werden) erstellt. Jede Zeile wird als Objekt mit den dazugehörigen Spalten dargestellt.

```
Date,Value
2010-12-31,4220.717
2009-12-31,4352.729
2008-12-31,5555.505
2007-12-31,5067.794
```

```
[
  {
    "Date": "2010-12-31",
    "Value": "4220.717"
  },
  {
    "Date": "2009-12-31",
    "Value": "4352.729"
  },
  {
    "Date": "2008-12-31",
    "Value": "5555.505"
  },
  {
    "Date": "2007-12-31",
    "Value": "5067.794"
  }
]
```

Abbildung 2.2: Konvertierung von CSV zu JavaScript Objekten. Links: CSV. Rechts: JSON.

Formatierung

Die Applikation benötigt nun Anweisungen, um den Datensatz (als Array) zu formatieren, damit er im Diagramm verwendet werden kann. Der Prozess muss folgende Aufgaben erledigen (diese Strategie wurde selber entwickelt):

- (Zeichenstring-) Elemente gegebenfalls in JavaScript-Objekte umwandeln
- Falls mehrere Datensätze vorhanden sind, diese *mergen*, also in einen einzigen Array zusammenfassen
- Den gesamten gemergten Datensatz nach der unabhängigen Variable aufsteigend sortieren

Umwandlung zu JavaScript-Objekten. Das CSV-Format unterscheidet nicht zwischen Datentypen. Alle Werte in CSV-Dateien sind Zeichenstrings.

Im Beispiel in der Abbildung 2.2 rechts, Zeile 3, wird für das erste Objekt im Array das Attribut mit dem Namen „Date“ definiert. Der Datentyp ist hier ein Zeichenstring; damit ist eine Umwandlung in das JavaScript Date-Objekt, das ein Datum darstellt, sinnvoll: Das JavaScript Date-Objekt beherrscht viele Funktionen, wie zum Beispiel die Ausgabe der Anzahl Millisekunden, die seit dem 1. Januar 1970 vergangen sind [17]. Dies ist beim Vergleichen von verschiedenen Date-Objekten nützlich. Das Date-Objekt ist zum Beispiel auch fähig, das Datum in einem Format auszugeben, das den lokalen Gegebenheiten entspricht: 28.10.2015 (Schweiz), 10/28/2015 (USA).

Zahlen, wie in Abbildung 2.2 rechts, Zeile 4, im Attribut mit dem Namen „Value“ definiert, werden zunächst von JavaScript als Zeichenstring behandelt. Diese Strings müssen in Zahlen-Objekte umgewandelt werden, denn nur mit Zahlen-Objekten können Rechenoperationen durchgeführt werden.

Merging von Datensätzen. Oft werden mehrere Datensätze in der Applikation geladen. Ein Beispiel ist der CO_2-Ausstoss von Ländern: Für jedes Land wird eine separate CSV-Datei geladen und in einen JavaScript-Array umgewandelt.

Es wurde entschieden, alle geladenen Datensätze (Arrays) in einen Datensatz (Array) zusammenzufassen (*merge*), weil die Umsetzung aus Sicht der Programmierlogik einfacher ist.

Da Spalten von verschiedenen Datensätzen meist mit gleichem Namen beschriftet sind, könnte man nach dem Merge die Spalten nicht unterscheiden (Abbildung 2.3 links). Darum wird die Beschriftung aller Spalten der abhängigen Variablen durch eine eindeutige Identifikation ersetzt (Abbildung 2.3 rechts). Die eindeutige Identifikation wird durch den Spaltennamen und die URL des Datensatzes generiert: Dem Spaltennamen werden ein Rautenzeichen und die URL angehängt. Dies ermöglicht, dass man trotz Merge die Objekte dem ursprünglichen Datensatz zuordnen kann.

In der Abbildung 2.3 wurden zwei Datensätze mit den Dateinamen ch-co2.csv und af-co2.csv gemergt. Im rechten Beispiel wurden die Spaltennamen der abhängigen Variablen durch die eindeutige Identifikation ersetzt.

```
[
  {
    "Date": "2010-12-31",
    "Value": "4220.717"
  },
  {
    "Date": "2009-12-31",
    "Value": "4352.729"
  },
  {
    "Date": "2010-12-31",
    "Value": "1320.717"
  },
  {
    "Date": "2009-12-31",
    "Value": "7353.129"
  }
]
```

```
[
  {
    "Date": "2010-12-31",
    "Value#ch-co2.csv": "4220.717"
  },
  {
    "Date": "2009-12-31",
    "Value#ch-co2.csv": "4352.729"
  },
  {
    "Date": "2010-12-31",
    "Value#af-co2.csv": "1320.717"
  },
  {
    "Date": "2009-12-31",
    "Value#af-co2.csv": "7353.129"
  }
]
```

Abbildung 2.3: Demonstration der Merge-Strategie und Anwendung der ID-Generierung. Links: Gemergter Datensatz, ohne eindeutige IDs. Rechts: Gemergter Datensatz, mit eindeutigen IDs.

Sortieren des gemergten Datensatzes. Der Datensatz wird nach der unabhängigen Variable ansteigend sortiert, damit die Berechnung von Interpolationen ermöglicht wird.

Datensatzspezifische Filteringkonfiguration und Hard Coding

Die Applikation soll fähig sein, mehrere Formate von CSV-Datensätzen verarbeiten zu können. Der Benutzer sollte die Anweisungen zum Filtering anpassen können.

Die Implementierung dieser Anweisungen zum Filtering (Spezifikation der URLs, abhängige und unabhängige Variablen, Datentypen der Spalten des Datensatzes) kann auf zwei Arten durchgeführt werden:

- Die Anweisungen zum Filtering sind im Sinne des *Hard Coding* implementiert;
- Die Anweisungen zum Filtering sind in einer *Konfiguration* abgelegt (*Soft Coding*).

Hard Coding. Die Anweisungen (Konfiguration) zum Filtering sind direkt im Programmcode abgelegt. Falls Datensätze mit anderen URLs, Spaltennamen oder Datentypen in der Applikation verwendet werden sollen, so müssen die Anweisungen direkt im Programmcode entsprechend angepasst werden.

Soft Coding. Beim Soft Coding sind die Anweisungen (Konfiguration) für das Filtering in einer externen Ressource abgelegt, zum Beispiel in einer Datei oder Datenbank. Die Applikation liest die Konfiguration und führt das Filtering dementsprechend aus. Dem Benutzer ist es so möglich, die Funktionsweise der Applikation anzupassen. Soft Coding wird in dieser Applikation verwendet.

meta.json

Die Applikation verwendet als Konfiguration eine Datei im JSON-Format, die vor dem Filtering in ein JavaScript-Objekt umgewandelt werden kann. In Abbildung 2.4 ist ein Beispiel der Konfigurationsdatei meta.json aufgelistet:

- `datasets` (Zeile 2-23) beschreibt einen Array für Datensätze;
- `url` (Zeile 4) gibt die URL des Datensatzes an;
- `config` (Zeile 5-21) ist ein Array, in dem die Spalten des Datensatzes konfiguriert werden;
- `row` (Zeile 7, 14) gibt die zu konfigurierende Spalte an;
- `type` (Zeile 8, 15) hat entweder den Wert `index` oder `value`. `index` bedeutet, dass die Spalte eine unabhängige, `value` bedeutet, dass die Spalte eine abhängige Variable ist;
- `data_type` (Zeile 9, 16) gibt den Datentyp der Spalte an, damit sie in das entsprechende JavaScript-Objekt umgewandelt werden kann. Er kann den Wert `Date` oder `Number` haben

- `date_format` (Zeile 10) gibt bei Spalten mit Datentyp `Date` das Format des Datums an, damit es geparst werden kann;
- `name` (Zeile 11, 17) setzt einen Namen die Spalte, die im Diagramm angezeigt werden soll;
- `activated` (Zeile 18) gibt an, ob der Datensatz standardmässig im Diagramm angezeigt werden soll. Dieses Attribut spielt besonders eine Rolle bei Diagrammen, die mehrere Datensätze darstellen;
- `unit` (Zeile 19) stellt die Einheit einer abhängigen Variable dar. Sie wird als Information im Diagramm angezeigt.

```
1  {
2    "datasets": [
3      {
4        "url":"data/WWDI-AFG_EN_ATM_CO2E_KT.csv",
5        "config": [
6          {
7            "row": "Date",
8            "type": "index",
9            "data_type": "Date",
10           "date_format": "%Y-%m-%d",
11           "name": "Datum"
12         },
13         {
14           "row": "Value",
15           "type": "value",
16           "data_type": "Number",
17           "name": "Afghanistan",
18           "activated": true,
19           "unit": "kt"
20         }
21       ]
22     }
23   ]
24 }
```

Abbildung 2.4: **meta.json**: Beispiel der Konfigurationsdatei, welche die Applikation verwendet.

2.2.3 Mapping

Das Mapping ist der Prozess der Umwandlung von aufbereiteten Daten zu Geometriedaten, es ist das Kernstück der Diagrammerstellung. Es wird eine *Daten-zu-Geometrie-Abbildung* realisiert [1, Kapitel 2.1].

In der Applikation, die in dieser Arbeit entwickelt wird, stellen die Geometriedaten die SVG-Elemente wie zum Beispiel Rechtecke, Kreise, Linien, Text dar.

Die Programmlogik des Mappings unterscheidet sich grundlegend bei jedem Diagrammtyp. In dieser Arbeit werden mehrere Typen von Diagrammen entwickelt, der Prozess des Mappings wird daher für jeden Typ separat behandelt.

2.2.4 Rendering

Das Rendering ist der letzte Schritt der Diagrammerstellung, bei dem die Abbildung der Geometriedaten in Bilddaten erfolgt [1, Kapitel 2.1].

In unserem Falle sind Bilddaten das Diagramm, das im Browser angezeigt wird. Es ist die Aufgabe des Browsers, die Geometriedaten (SVG-Elemente) mit Darstellungsanweisungen (CSS) in Bilddaten (Pixel, angezeigte Grafik) zu rendern.

2.3 Information-Seeking Mantra

Als Startreferenz für die Entwicklung des interaktiven Diagrammes bieten sich die von Ben Shneiderman begründeten Prinzipien für das Design graphischer Benutzeroberflächen an, das „*Information-Seeking Mantra*“. Die Weise, wie der Benutzer mit der Oberfläche interagiert, hat Shneiderman [2] festgelegt:

- Überblick („*Overview first*“);
- Zoomen und Filtern („*zoom and filter*“);
- Details auf Abruf („*then details-on-demand*“).

Überblick. Der Benutzer verschafft sich einen Überblick über die gesamte Oberfläche des Programms.

Zoom. Zur besseren Betrachtung vergrössert der Benutzer die Ansicht, sodass die betreffenden Elemente grösser angezeigt werden.

Filter. Die Filter-Funktion ermöglicht dem Benutzer, gewisse Elemente oder Elementgruppen je nach Interesse ein- oder auszublenden. Diese Filter-Funktion der Benutzeroberfläche ist nicht zu verwechseln mit dem Filtern von Datensätzen, das Rohdaten in aufbereitete Daten umwandelt.

Details auf Abruf. Falls ein Element den Benutzer besonders interessiert, besteht die Möglichkeit, dass zusätzliche relevante Informationen zum Element angezeigt werden können.

Zusätzlich formulierte Shneiderman drei weitere Schritte:

Zusammenhänge betrachten. Die Zusammenhänge zwischen den verschiedenen Elementen können im Programm betrachtet werden. Diesen Schritt umzusetzen ist nur bei wenigen Benutzeroberflächen sinnvoll, zum Beispiel bei der Darstellung von Baumdiagrammen oder anderen Diagrammen mit hierarchischen Daten.

Verlauf. Die Interaktionen des Benutzers mit der Programmoberfläche werden aufgezeichnet. Dadurch können Interaktionen rückgängig gemacht werden.

Extraktion. Damit wird die Extraktion der *Query-Parameter* (wie angewandte Filter, Verlauf, Zoomstufe) und der durch die Interaktion bereits definierten Elementgruppen („*subcollections*“) erlaubt.

Im Diagramm werden nur die Methoden *Überblick*, *Zoom*, *Filter* und *Details auf Abruf* umgesetzt. Das Mantra wurde allgemein für Benutzeroberflächen von Programmen beschrieben, in unserer Applikation, einem interaktiven Diagramm, macht aber die Umsetzung der Schritte *Zusammenhänge betrachten*, *Verlauf*, *Extraktion* in Bezug auf die Funktion des Diagramms keinen Sinn:

- Punktdiagramme oder Liniendiagramme werden nicht dazu verwendet, Daten in Hierarchieform darzustellen.
- Die Interaktionen im Diagramm sind zu banal, als dass eine Rückgängig-Funktion von Nutzen wäre.

- Die Extraktion und der Export von Query-Parametern des Diagramms ist zwar theoretisch umzusetzen, ist jedoch von minimaler praktischer Bedeutung.
- Zur Extraktion von „*subcollections*“ aus einem Datensatz sollte kein interaktives Diagramm verwendet werden. Eine Datenverarbeitungsapplikation ist für diese Aufgabe angemessener, da exakte Parameter zur Extraktion bestimmt werden können.

2.4 Zweidimensionales Punktdiagramm

Das zweidimensionale Punktdiagramm ist weit verbreitet, darum sind sich Betrachter an diese Darstellung gewöhnt. Oft werden die Punkte im Diagramm durch eine Linie verbunden, was den Verlauf der abgebildeten Datenwerte verdeutlicht, besonders in Medien, zum Beispiel für die Darstellung von Börsenkursen.

Aus diesem Grund wurde als erstes Beispiel das zweidimensionale Punktdiagramm (beziehungsweise das Liniendiagramm, falls Linien hinzugefügt werden) ausgewählt.

2.4.1 Applikation

Der Screenshot in Abbildung 2.5 zeigt die entwickelte Applikation. Es sind Datensätze [14] zum CO_2-Ausstoss von 15 Ländern vorhanden, drei davon sind im Diagramm eingeblendet. Der Name des Tests in der Applikation lautet `layout`.

Achsen. Es wurden Achsen mit linearer Skala für die Applikation gewählt. Beschriftungen und Markierungen, sogenannte *Ticks*, sind vorhanden. Die Ticks auf der Ordinatenachse (y-Achse) wurden in Richtung Mitte des Diagramms verlängert, so dass ein Gitter entsteht.

Falls in einem Diagramm mehrere Datensätze gleichzeitig dargestellt werden sollen, wie es der Fall in Abbildung 2.5 ist, so müssen die abgebildeten Werte die gleiche Einheit (zum Beispiel Meter, Kilogramm, Franken) besitzen. Bei der gleichzeitigen Darstellung von mehreren Datensätzen mit verschiedenen Einheiten in einem Diagramm müsste für jede vorhandene Einheit eine eigene Achse und Skala gezeichnet werden.

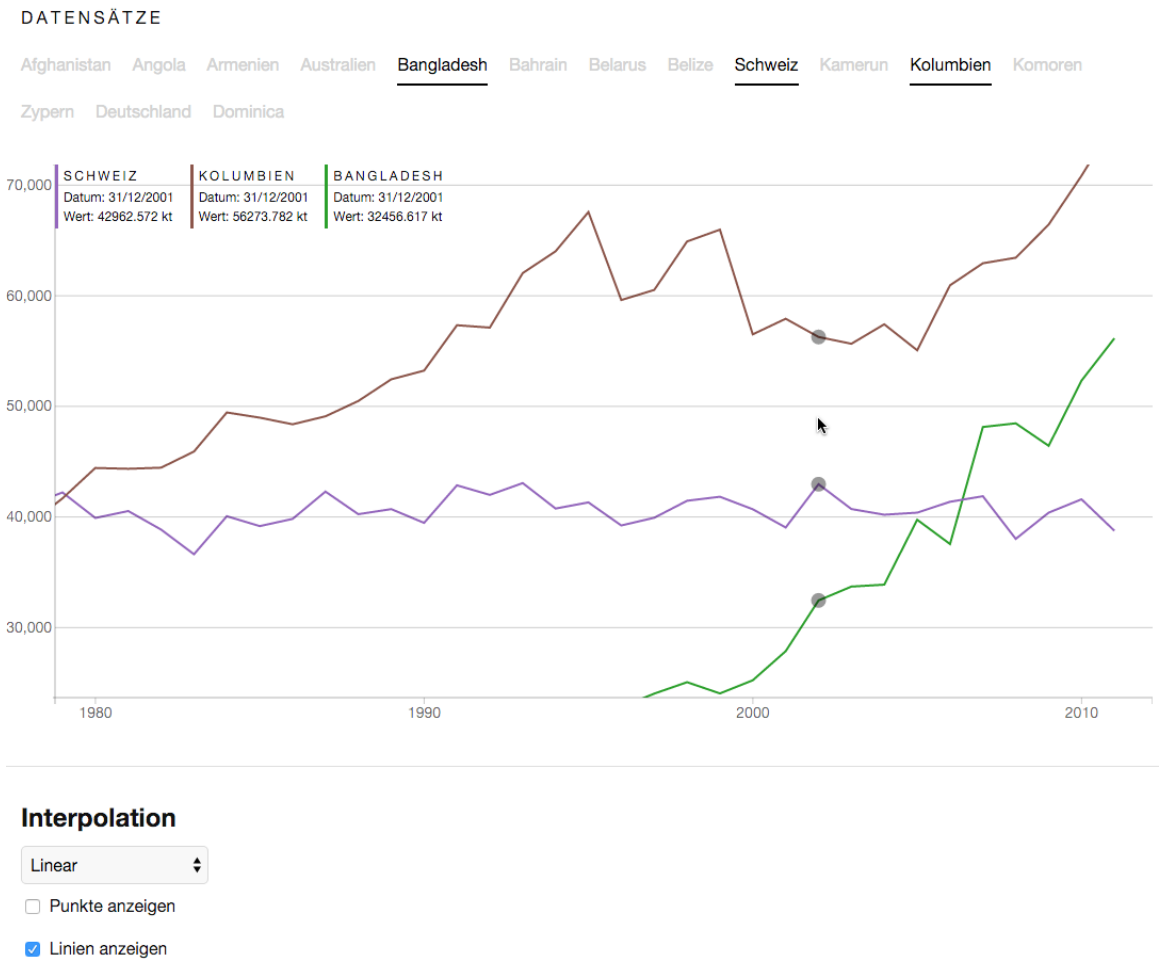

Abbildung 2.5: Screenshot der Oberfläche der Applikation (zweidimensionales Liniendiagramm / Punktdiagramm)

Datenpunkte. Die Applikation bietet an, die Daten als Punkte und/oder verbunden durch eine Linie anzuzeigen. In Abbildung 2.5 unten wird durch Kontrollkästchen ermöglicht, die Anzeige zu verändern.

Layout und Textformatierung. Für den Schritt *Übersicht* sind das Layout und auch die Formatierung des Textes von Bedeutung. Eine klare Oberflächenstruktur nach modernen Minimalismus (nach Designtrends wie *Flat Design* oder *Material Design*) wurde geschaffen, bei der der Inhalt im Vordergrund steht. „Benutzer betrachten keine Details, sie benutzen Details.“ [18].

Zur besseren Orientierung des Benutzers wird hier das Prinzip von „*pop*“ und „*unpop*“ von Text, das von Erik Kennedy [19] beschrieben wurde, berücksichtigt: Durch die Anpassung der Typographie kann ein Text hervorgehoben beziehungsweise in den Hintergrund gestellt werden. Folgende Eigenschaften von Texten im Diagramm werden in der Applikation entsprechend der Wichtigkeit angepasst:

- Grösse (grösser bzw. kleiner);
- Farbe (grösserer bzw. kleinerer Kontrast);
- Schriftstärke (fetter bzw. leichter);
- Schriftart (Grossschreibung bzw. Kleinschreibung);
- Laufweite (kleiner bzw. grösser);
- Unterstreichung (unterstrichen bzw. nicht unterstrichen).

Linien. Der Verlauf der abgebildeten Datenwerte wird verdeutlicht, indem die Punkte im Diagramm durch eine Linie verbunden werden. Die lineare Interpolation wurde durch ein selbstgeschriebenes Modul implementiert. Weitere Interpolationen, wie der *Kubisch Hermitescher Spline* oder *Basis-Spline*, wurden mittels D3 implementiert und können ebenfalls auf das Diagramm angewendet werden.

Verschiedene Farben von Linien ermöglichen die Zuordnung von Datensätzen. Die den Datensätzen zugewiesenen Farben werden in der Detailanzeige aufgeführt.

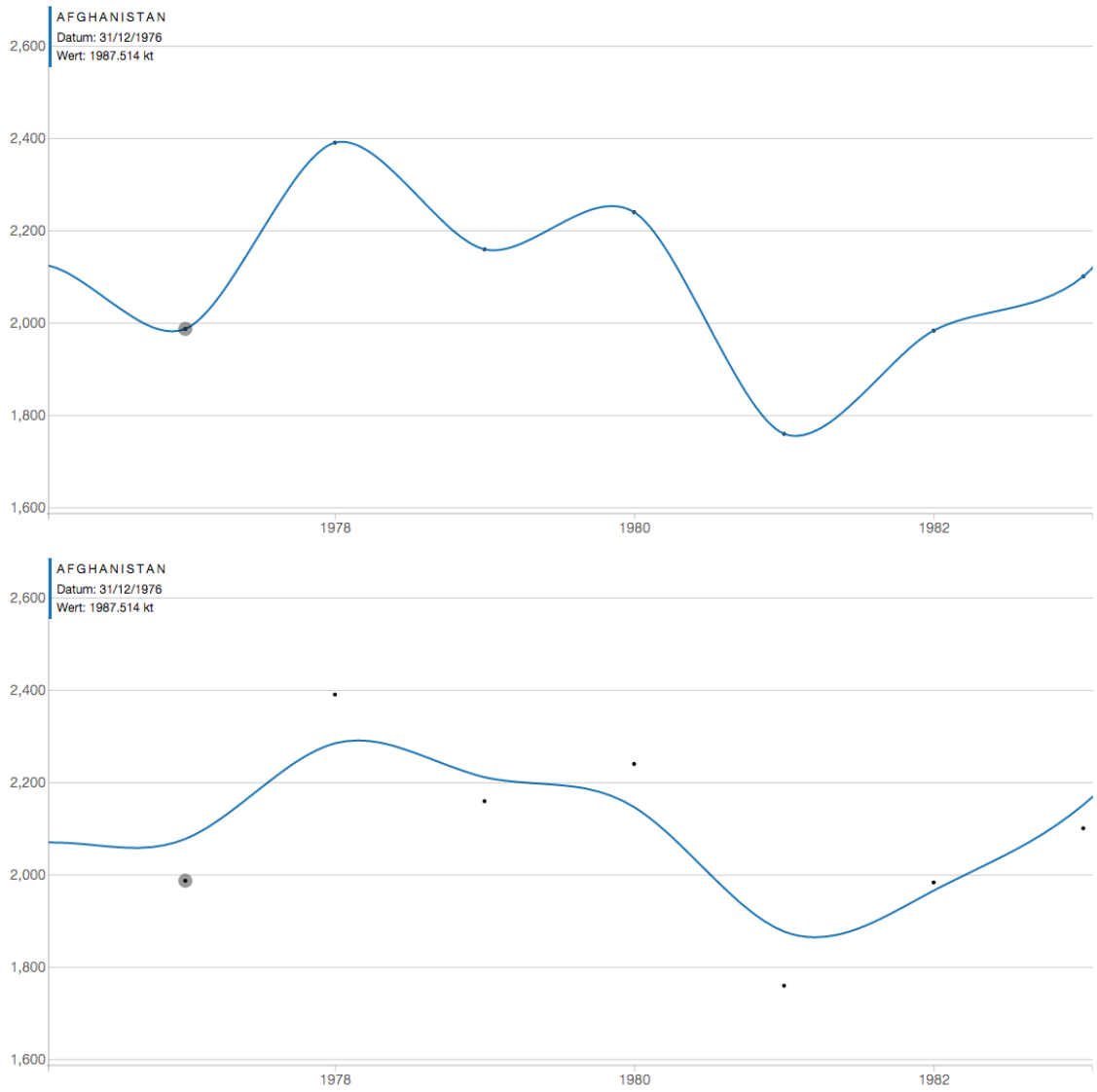

Abbildung 2.6: Beispiele von Interpolationen im Diagramm. Von oben nach unten: Kubisch Hermitescher Spline, Basis-Spline.

Zoom. Der Schritt *Zoom* des *Information-Seeking Mantra* wurde im Diagramm umgesetzt. Es kann durch Scrollen gezoomt werden; die Skalierungen der Achsen werden verändert. Die Programmlogik des Zoomens wurde mittels D3 implementiert.

Es gibt zahlreiche interaktive Diagramme, in denen nur der Zoom durch Veränderung der Skalierung der Abszisse möglich ist. Dabei wird nach jedem Zoom-Vorgang auch die Skalierung der Ordinate neu angepasst oder überhaupt nicht verändert.

Diese Zoom-Strategie verwirrt den Benutzer, weil sich die Skalierungen der Achsen nicht gleichmässig beim Zoom verändern: Er nimmt ein Dehnen bzw. Zusammendrücken des Diagramms wahr. Der Benutzer ist es sich gewöhnt, dass die beiden Skalierungen stets im gleichen Verhältnis stehen; dies ist auch beim Zoom bei Smartphones üblicherweise der Fall. Deshalb wurde diese Zoomstrategie bevorzugt und umgesetzt.

Datensatzauswahl. In diesem Beispiel (Abbildung 2.5) können Linien ein- und ausgeblendet werden, die Datensätzen für 15 Länder entsprechen. Dieser Vorgang gehört zum Schritt *Filtern*. Aktivierte, angezeigte Datensätze werden nach den Prinzipien von Kennedy [19] hervorgehoben, nicht aktivierte, versteckte Datensätze sind in den Hintergrund gestellt.

Tooltip und Detailanzeige. Beim Tooltip wird der der Maus am nächsten liegende Datenpunkt jedes Datensatzes markiert und weiter in der Detailanzeige beschrieben: In der Detailanzeige werden genaue x- und y-Werte mit Einheit angezeigt. Dieser Vorgang entspricht dem Schritt *Details auf Abruf*.

Dynamik der Applikation. Der Screenshot dieses Beispiels (Abbildung 2.5) ist das Resultat der Anzeige der Applikation, die für bestimmte Datensätze konfiguriert ist. Elemente des Diagramms, wie der Wertebereich der Achsen, die Datensatzauswahl, der Tooltip und die Detailanzeige, werden dynamisch generiert, so dass diese Elemente ohne weitere Einstellung auch für andere Datensätze funktionieren.

2.5 Dreidimensionales Punktdiagramm

Falls Datensätze mit einer unabhängigen und zwei abhängigen Variablen dargestellt werden sollen, eignen sich zweidimensionale Punkt-/Liniendiagramme weniger für die Darstellung: Sie können mit zwei Achsen Datensätze mit nur einer unabhängigen und einer abhängigen Variable darstellen. Anstelle des zweidimensionalen Punktdiagramms kann jedoch ein dreidimensionales Punktdiagramm verwendet werden, das drei Achsen besitzt.

Beispiele für Datensätze, die in dreidimensionalen Punktdiagrammen dargestellt werden können:

- Zwei Attribute im Laufe der Zeit, zum Beispiel BIP und Arbeitsplätze eines Landes.
- Drei Attribute, die in einem bestimmten Verhältnis zueinander stehen, zum Beispiel Geschwindigkeit, Luftwiderstand und Oberfläche.

Es wurde entschieden, eine weitere Applikation zu entwickeln, die einen Datensatz mit einer unabhängigen und zwei abhängigen Variablen in einer 3D-Applikation anzeigt. Dazu wurden Daten der World Bank [14] zur Bevölkerung und Arbeitsplätzen der Schweiz im Verlauf der Zeit verwendet.

2.5.1 Applikation

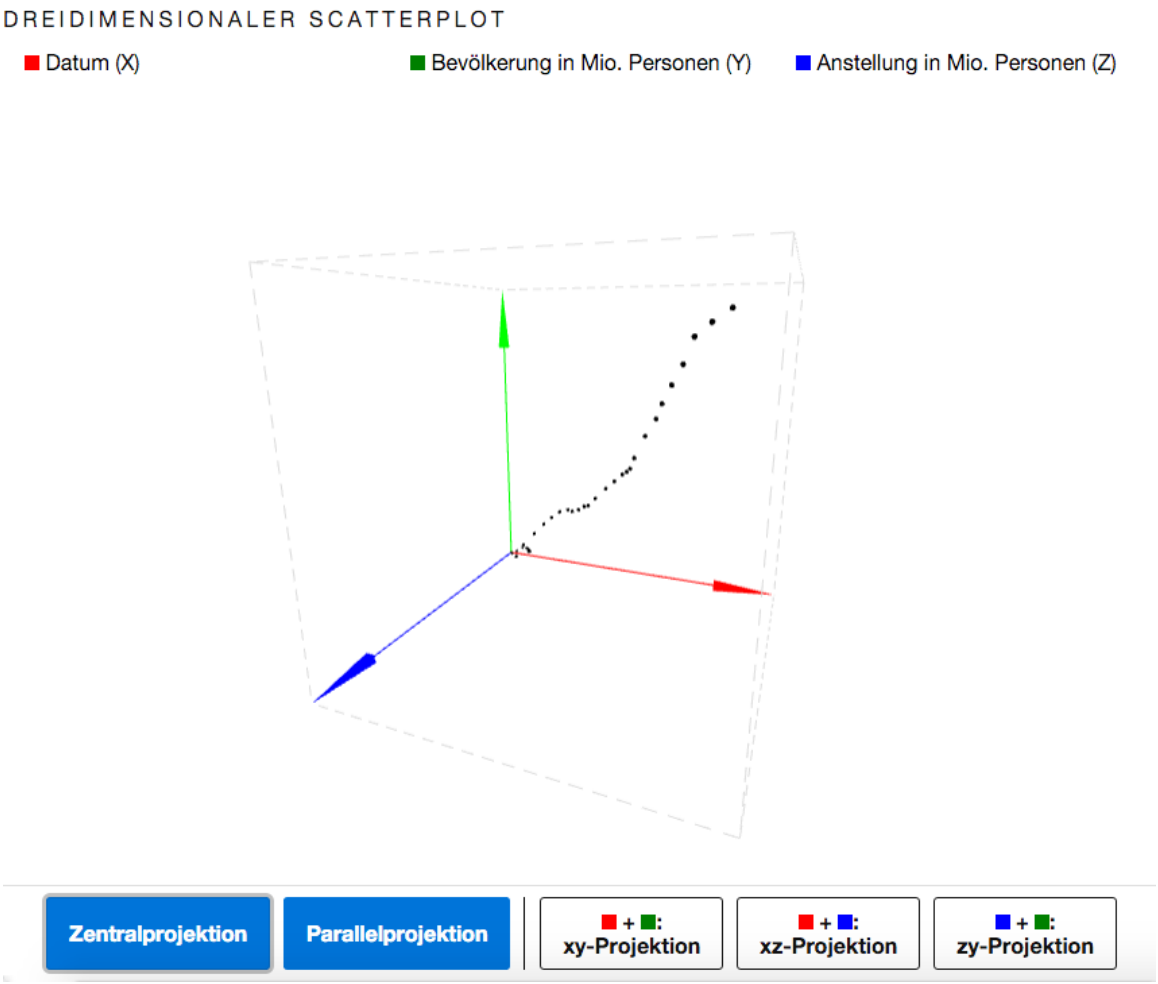

Abbildung 2.7: Oberfläche der Applikation (dreidimensionales Punktdiagramm), Zentralprojektion

Abbildung 2.7 zeigt die Oberfläche der entwickelten Applikation. Der Name des Tests in der Applikation lautet `3d`. Die Programmbibliothek *three.js* [7] wurde verwendet; three.js ermöglicht die Entwicklung von 3D-Applikationen mit JavaScript im Browser.

Achsen. Als Achsen wurden verschiedenfarbige Pfeile (Abbildung 2.8) mit Richtungsvektoren $\vec{a}_x$, $\vec{a}_y$, und $\vec{a}_z$ erstellt.

$$\vec{a}_x = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \text{ (rot)} \qquad \vec{a}_y = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{ (grün)} \qquad \vec{a}_z = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \text{ (blau)}$$

Abbildung 2.8: Die Richtungsvektoren der Achsen im dreidimensionalen Punktdiagramm.

Die Beschriftung und Farben der Achsen werden in der Applikation über dem Diagramm angezeigt. Die erste Beschriftung (rot) steht für die unabhängige Variable, die zweite und dritte Beschriftung (grün und blau) für die abhängigen Variablen.

Punkte. Die Datenpunkte werden an den drei Achsen abgetragen und als schwarze Kugeln im Raum abgebildet.

Würfel. Ein Würfel wird mit gestrichelten grauen Linien um das Diagramm dargestellt. Es soll dem Benutzer helfen, sich im dreidimensionalen Raum zu orientieren.

Rotation. Das dreidimensionale Punktdiagramm ist mit der Maus rotierbar. Die Kamera bleibt während der Rotation stets auf die Mitte des Diagramms gerichtet. Die Rotation ermöglicht eine bessere Erkundung des Datensatzes und bietet die Ansicht aus verschiedenen Winkeln dar.

Eine Funktion für das Zoomen wurde nicht implementiert, weil sie sich als kontraproduktiv herausstellte: Die Orientierung des Benutzers geht schnell verloren, zum Beispiel wenn zu weit herausgezoomt wird, sodass das Diagramm nicht mehr sichtbar ist, oder wenn sich die Kamera innerhalb der Punktewolke befindet.

2.5.2 Reduktion auf zweidimensionale Punktdiagramme durch Projektion

Eine besondere Erkenntnis bei der Entwicklung dieser Applikation ist das Erkennen des Potentials von Parallelprojektionen innerhalb des dreidimensionalen Punktdiagramms.

Das Punktdiagramm kann durch Verschieben der Kamera und Verwendung der Parallelprojektion auf ein zweidimensionales Punktdiagramm reduziert werden.

Zum Beispiel führt die Projektion der zy-Fläche (Abbildung 2.9 rechts) zur Darstellung eines zweidimensionalen Punktdiagramms: Die *neue Abszisse*[1] ist die y-Achse, die *neue Ordinate*[2] die z-Achse. Der Effekt kann in diesem Beispiel erzielt werden, wenn die Kamera auf negativer x-Achsen-Position, halber y-Achsen-Position und halber z-Achsen-Position ist und in die Richtung der Mitte des Diagramms ausgerichtet ist. Kurz gesagt, die Kamera ist in die Richtung der x-Achse ausgerichtet.

[1] Mit dem Begriff „*neue Abszisse*“ ist die Abszisse des neu dargestellten, zweidimensionalen Punktdiagramms gemeint.

[2] Mit dem Begriff „*neue Ordinate*“ ist die Ordinate des neu dargestellten, zweidimensionalen Punktdiagramms gemeint.

Abbildung 2.9 veranschaulicht die Funktionsweise der xy-, xz- und zy-Projektion.

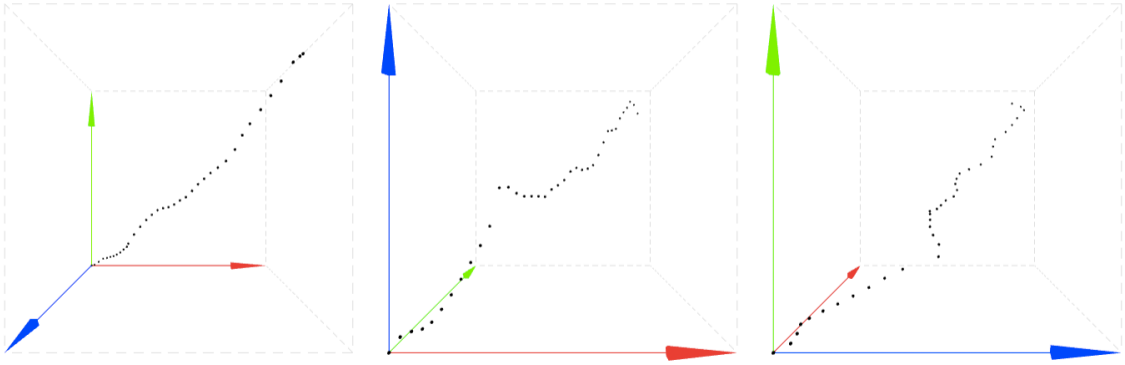

Abbildung 2.9: Funktionsweise der Projektionen, von links nach rechts: xy, xz und zy.

Die Gleichungen 2.1 erklären die Funktionsweise der Projektion: Sei S der Schnittpunkt der Abszissenachse (x), Ordinatenachse (y) und Applikatenachse (z), a die Länge einer Achse, M der Mittelpunkt des Diagramms und P_{xy}, P_{xz}, P_{zy} die Position der Kamera für die entsprechende Projektion:

$$\begin{aligned} S &= (0|0|0) \\ M &= (\frac{a}{2}|\frac{a}{2}|\frac{a}{2}) \\ P_{\text{xy}} &= (\frac{a}{2}|\frac{a}{2}|a+a) \\ P_{\text{xz}} &= (\frac{a}{2}|-a|\frac{a}{2}) \\ P_{\text{zy}} &= (-a|\frac{a}{2}|\frac{a}{2}) \end{aligned} \tag{2.1}$$

Somit stehen je zwei Achsen bei P_{xy}, P_{xz} oder P_{zy} senkrecht, die verbleibende Achse parallel zur Kamerarichtung; in Abbildung 2.10 werden die neue Abszisse und Ordinate des projizierten zweidimensionalen Punktdiagramms und die Kamerarichtung nach Projektion aufgelistet.

Projektion	**neue Abszisse**	**neue Ordinate**	**Kamerarichtung**
xy	Abszissenachse (x)	Ordinatenachse (y)	Entgegen der Applikatenachse (z)
xz	Abszissenachse (x)	Applikatenachse (z)	In Richtung Ordinatenachse (y)
zy	Applikatenachse (z)	Ordinatenachse (y)	In Richtung Abzissenachse (x)

Abbildung 2.10: Beschreibung des projizierten zweidimensionalen Punktdiagramms

In Abbildung 2.11 wird die Projektion der xz-Ebene angezeigt. Beim linken Beispiel wird die *Zentralprojektion* verwendet. Die Zentralprojektion wird in den meisten 3D-Applikationen verwendet. Die Geraden verlaufen durch einen festen Punkt. Das Auge verwendet ebenfalls die Zentralprojektion: Strahlen werden so auf der Netzhaut abgebildet.

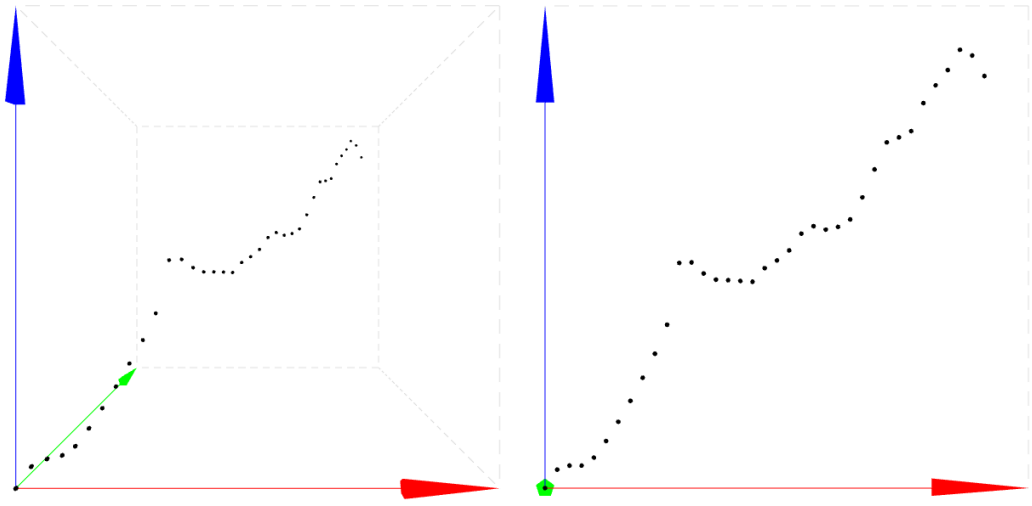

Abbildung 2.11: Projektion der XZ-Ebene. Links: Zentralprojektion. Rechts: Parallelprojektion.

Die Zentralprojektion hat den Nachteil, dass sie Objekte verzerrt, je nachdem, wie weit sie entfernt sind. Sie ist für unsere Zwecke (der Reduktion des dreidimensionalen Punktdiagramm auf ein zweidimensionales) nicht geeignet.

Bei der *Parallelprojektion* hingegen werden Punkte auf einer Ebene abgebildet und somit nicht verzerrt, falls sie weiter entfernt von der Projektionsfläche liegen. Beim rechten Beispiel in Abbildung 2.11 wird die Parallelprojektion angewandt, somit ist das dreidimensionale Punktdiagramm auf ein zweidimensionales Punktdiagramms reduziert worden.

Es ist in der Applikation möglich, zwischen der Zentralprojektion und Parallelprojektion sowie zwischen der xy-, xz- und zy-Projektion umzuschalten. Der Benutzer kann sich so besser mit dem Datensatz auseinandersetzen, indem er einfach zwischen dreidimensionalem und zweidimensionalem Punktdiagramm navigieren kann.

Damit der Benutzer die Orientierung behält, wird die Kamerabewegung zwischen P_{xy}, P_{xz} und P_{zy} animiert. Dazu wird die JavaScript-Bibliothek *tween.js* [20] verwendet.

2.6 Vergleich eines statischen und dynamischen Diagramms

Man kann sich nun die Frage stellen, wieso sich nun dynamische besser als statische Diagramme in der Praxis für die Darstellung eignen. Um die Vorteile von dynamischen Diagrammen anschaulicher zu demonstrieren, wird zuerst ein Beispiel eines statischen Diagramms gesucht, das in der Praxis verwendet wird. Anschliessend wird ein dynamisches Diagramm mit der Applikation generiert, das den gleichen Datensatz des statischen Diagramms verwendet: Die zwei Diagramme sollen einen äquivalenten Datensatz darstellen.

Als Beispiel des statischen Diagramms wurde das Diagramm des Wechselkurses von Euro in Schweizer Franken von der Neuen Zürcher Zeitung vom 8.11.2015 (Abbildung 2.12 rechts) gewählt, weil der dargestellte Datensatz bekannt ist: Datensätze von Börsen sind öffentlich, frei und in vielen Formaten verfügbar und können deshalb leicht in der Applikation verwendet werden.

Abbildung 2.12: Vergleich des dynamischen (links) und statischen Diagramms (rechts, NZZ 8.11.15) bei der Darstellung eines äquivalenten Datensatzes.

Der Datensatz wird in beiden Beispielen in Abbildung 2.12 mittels eines zweidimensionalen Liniendiagramm dargestellt. Man kann erkennen, dass es sich um denselben Datensatz handelt.

2.6.1 Interaktion

Verschiebung und Zoom. Beim dynamischen Diagramm ist es möglich durch Verschieben und Zoomen beliebige Bereiche des Diagramms vergrössert anzuzeigen. Somit kann der Benutzer zum Beispiel den Verlauf des Wechselkurses in einem Zeitraum von drei Tagen einsehen.

Im dynamischen Diagramm ist es so möglich, den genauen Verlauf des Wechselkurses während des Währungscrashs am 15. Januar und der folgenden Tagen zu analysieren; ein solches Vorgehen wäre beim statischen Diagramm nicht möglich.

Tooltip und Details auf Abruf. Der Wert des Wechselkurses an einem bestimmten Tag kann abgerufen werden, indem der Benutzer mit der Maus in die Nähe eines bestimmten Datenpunktes fährt. In Bezug auf diesen Datensatz kann zum Beispiel das Datum ermittelt werden, an dem der Wechselkurs am tiefsten war.

Beim statischen Diagramm können keine genauen Werte abgelesen, sondern nur mit Hilfe der Achsenbeschriftungen abgeschätzt werden.

2.6.2 Dynamik

Da bei jedem Aufruf des dynamischen Diagramms der Datensatz neu geladen wird und anschliessend dargestellt wird, ist es möglich, den Datensatz stets zu aktualisieren. In dieser Applikation würde das zum Beispiel bedeuten, dass jeden Tag der neuste Datensatz der Börse verwendet werden und somit immer der aktuellste Kurs im Diagramm dargestellt werden würde.

Dies wurde nicht in dieser Applikation umgesetzt, jedoch wird dieses Prinzip zum Beispiel bei der interaktiven Kursanzeige von Yahoo [21] verwendet: Der Kurs wird jede Minute automatisch aktualisiert und angezeigt.

Da dynamische Diagramme im erweiterten Sinne Webapplikationen sind, die in einem Browser ausgeführt werden, können diese auch im Nachhinein mit einer praktisch unendlichen Anzahl Funktionen erweitert werden, da bei jedem Aufruf der Seite der Quellcode vom Server heruntergeladen wird.

Kapitel 3

Schlusswort

Durch Umsetzung von verschiedenen Interaktionsmethoden an selbstentwickelten Diagrammen konnte die Analyse und das Verständnis gegenüber der statischen Version verbessert werden.

Neben der Dokumentation des Entwicklungsprozesses und der verwendeten Technologien wurde für die interaktiven Diagramme eine optimale Datenverarbeitungsstrategie entwickelt (siehe „Merge“ und „meta.json“). Dabei wurde auch viel Wissen über JavaScript im Allgemeinen, JavaScript Buildsysteme (NPM, Gulp.js, Browserify), Open Source, Web Design (CSS, Minimalismus, Pop/Unpop) und über den Umgang mit Programmbibliotheken (D3, three.js, tween.js) erarbeitet.

Was den Entwicklungprozess der Applikation und das Verfassen des Codes betrifft, wurden folgende Erkenntnisse gewonnen: Bei Projekten mit einer solchen grossen Menge Code ist es sehr wichtig, Teile zu abstrahieren. Diese Abstraktion konnte mittels *Modulen* (Browserify) erzielt werden.

Beim zweidimensionalen Punkt-/Liniendiagramm wurden Möglichkeiten für die Interaktion mit Benutzeroberflächen entsprechend implementiert: Zoom, Tooltip, Detailanzeige und Datensatzauswahl. Es wurde so auch erreicht, dass Benutzer sich mit einer grossen Datenmenge effizienter auseinandersetzen können.

Zudem wurden Diagrammtechniken implementiert wie Achsen, Skalierungen und Interpolationen. Die Theorie hinter diesen Möglichkeiten und Techniken sind ebenfalls in der Arbeit dokumentiert worden.

Eine unkonventionelle Weise der Darstellung eines Datensatzes mit zwei abhängigen Variablen, das dreidimensionale Diagramm, wurde erläutert und als Applikation umgesetzt. Eine Methode wurde dazu entwickelt, welche die Benutzung des Diagramms produktiver machen sollte: Parallelprojektionen, die das dreidimensionale Punktdiagramm auf zweidimensionale Punktdiagramme reduzieren können.

Anhang A

Literaturverzeichnis

[1] Heidrun Schumann und Wolfgang Müller. *Visualisierung. Grundlagen und allgemeine Methoden*. 1. Aufl. Springer-Verlag Berlin Heidelberg, 2000. ISBN: 978-3-540-64944-1. DOI: 10.1007/978-3-642-57193-0.

[2] Ben Shneiderman. "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations". In: (1996). ISSN: 1049-2615.

[3] Michael Bostock et al. *Across U.S. Companies, Tax Rates Vary Greatly*. New York Times. 25. März 2015. URL: http://www.nytimes.com/interactive/2013/05/25/sunday-review/corporate-taxes.html (besucht am 04. 10. 2015).

[4] Michael Bostock, Shan Carter und Archie Tse. *Is It Better to Rent or Buy?* New York Times. 17. Juli 2015. URL: http://www.nytimes.com/interactive/2014/upshot/buy-rent-calculator.html (besucht am 04. 10. 2015).

[5] Feross Aboukhadijeh et al. *JavaScript Standard Style. One Style to Rule Them All*. 3. Okt. 2015. URL: https://github.com/feross/standard (besucht am 03. 10. 2015).

[6] Michael Bostock, Vadim Ogievetsky und Jeffrey Heer. "D3: Data-Driven Documents". In: *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011). URL: http://vis.stanford.edu/papers/d3.

[7] Ricardo Cabello et al. *three.js. JavaScript 3D library*. URL: https://github.com/mrdoob/three.js/ (besucht am 24. 10. 2015).

[8] Eric Schoffstall et al. *Gulp.js. Automate and enhance your workflow*. URL: http://gulpjs.com/ (besucht am 25. 10. 2015).

[9] *npm. A package manager for javascript*. npm, Inc. URL: https://github.com/npm/npm (besucht am 25. 10. 2015).

[10] James Halliday et al. *Browserify. Browserify lets you require('modules') in the browser by bundling up all of your dependencies*. URL: http://browserify.org/ (besucht am 25. 10. 2015).

[11] *BrowserSync. Time-saving synchronised browser testing*. JH. URL: http://www.browsersync.io/ (besucht am 25. 10. 2015).

[12] Mihai Bazon et al. *UglifyJS. JavaScript parser, mangler, compressor, beautifier library for NodeJS*. URL: https://github.com/mishoo/UglifyJS (besucht am 25. 10. 2015).

[13] Brent Jackson et al. *Basscss. Low-level CSS toolkit*. URL: http://www.basscss.com/ (besucht am 25. 10. 2015).

[14] *World Bank Open Data*. Free and open access to data about development in countries around the globe. The World Bank. URL: http://data.worldbank.org/ (besucht am 10. 16. 2015).

[15] Yakov Shafranovich. *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. RFC 4180. Internet Engineering Task Force. Okt. 2005. URL: https://tools.ietf.org/html/rfc4180 (besucht am 16. 10. 2015).

[16] Tim Berners-Lee, Larry Masinter und Michael McCahill. *Uniform Resource Locators (URL)*. RFC 1738. Internet Engineeing Task Force. Dez. 1994. URL: https://www.ietf.org/rfc/rfc1738.txt (besucht am 16. 10. 2015).

[17] *Date - JavaScript*. Mozilla Foundation. URL: https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Global_Objects/Date (besucht am 25. 11. 2015).

[18] Andrew Burton. *Modern minimalism is the right choice. Has flat design destroyed the role of the web designer?* 2. März 2015. URL: https://medium.com/@andrew.burton/modern-minimalism-in-web-design-is-the-right-choice-d91ed888a62b (besucht am 22. 10. 2015).

[19] Erik Kennedy. *7 Rules for Creating Gorgeous UI. A guide to visual aesthetics, written by a nerd*. 20. Nov. 2014. URL: https://medium.com/@erikdkennedy/7-rules-for-creating-gorgeous-ui-part-2-430de537ba96 (besucht am 22. 10. 2015).

[20] Soledad Penades et al. *tween.js. JavaScript tweening engine for easy animations, incorporating optimised Robert Penner's equations*. URL: https://github.com/tweenjs/tween.js/ (besucht am 25. 10. 2015).

[21] *EUR/CHF Währungsumrechner-Chart*. Yahoo. URL: https://de.finance.yahoo.com/echarts?s=EURCHF=X (besucht am 12. 11. 2015).

Anhang B

Abbildungsverzeichnis

Anhang C

Bestätigung der Eigenständigkeit

Der Unterzeichnete bestätigt mit Unterschrift, dass die Arbeit selbständig verfasst und in schriftliche Form gebracht worden ist, dass sich die Mitwirkung anderer Personen auf Beratung und Korrekturlesen beschränkt hat und dass alle verwendeten Unterlagen und Gewährspersonen aufgeführt sind.