Dokumentation zu `hbrs-thesis`

Modern und einfach

Version 1.0

30. August 2023

Inhaltsverzeichnis

Abbildungsverzeichnis

Codeverzeichnis

1 Vorwort

Diese Dokumentation ist nicht vollumfänglich. Das bedeutet, dass sie nicht auf alle Eigenheiten der einzelnen verwendeten Pakete eingeht. Für weitere und aktuelle Informationen empfehle ich immer die Recherche auf `https://www.ctan.org/`, wo alle Pakete mit ihren Dokumentationen hinterlegt sind.

In diesem Dokument werden die wichtigsten Befehle gezeigt und erklärt. Diese Dokumentation kann und sollte gerne durch die Hilfe der Community korrigiert und erweitert werden.

Viel Erfolg beim Schreiben!

2 Verwendung der LaTeX-Klasse

Um die Klasse als Vorlage für Dokumente zu verwenden, empfiehlt es sich, den kompletten Ordner `template` zu kopieren und entsprechend dieser Dokumentation zu verwenden. In diesem Kapitel werden die einzelnen Bestandteile des Ordners kurz erklärt. Die Verwendung der einzelnen Ordner und Dateien wird im späteren Verlauf des Dokuments näher erläutert.

2.1 Bestandteile des Templates

Die Konfiguration aller Informationen auf dem Deckblatt, sowie die Widmung können über die Datei `assets/utility/meta.tex` angepasst werden. Innerhalb des Ordners `assets/utility` finden sich außerdem noch eine Datei für Worttrennungen, die von LaTeXnicht korrekt erkannt werden (`hyphenation.tex`), Akronyme (`acronyms.tex`) und Glossareinträge (`glossary.tex`). `assets/utility/*`

In `assets/images` werden Bilder hinterlegt. Je nach Anzahl der Bilder im Dokument empfiehlt es sich, diese nach Kapitel zu sortieren. Im Ordner `assets/images` befindet sich bereits ein Ordner `titlepage`, welcher die Bilder für die Titelseite enthält. Diese sollten nicht gelöscht werden, da sonst die Titelseite und somit das Dokument nicht mehr gebaut werden kann. `assets/images/*`

Der Ordner `chapter` ist dafür gedacht, die Kapitel oder Abschnitte (je nach Verwendung der Dokumentklasse (siehe Kapitel 3)) aufzuteilen und die entsprechenden Dateien dort abzulegen. Diese Aufteilung bringt den Vorteil, dass Kapitel sehr schnell neu sortiert werden können. Zusätzlich enthalten die Dateien weniger Text und sind damit übersichtlicher. `chapter`

Zu einem wissenschaftlichen Dokument gehört auch Literatur. Die Informationen zu dieser Literatur werden als BibTeX oder BibLaTeX in der Datei `bibliography.bib` hinterlegt. Ich persönlich bevorzuge es die PDF-Dokumente lokal mit abzuspeichern. Mit lokalen Dateien habe ich die Möglichkeit Anmerkungen zu machen und die Dateien mit anderer Software besser zu durchsuchen (siehe `https://github.com/freedmand/semantra`). Die PDF-Dokumente kommen dann in den dieser Ordner `literature`. Je nach Literaturverwaltungssoftware können Einstellungen getroffen werden, um diese Dokumente direkt in der Software aufzurufen. `bibliography.bib` `literature`

Für den Bau des Dokuments mit `latexmk` wird die Datei `.latexmkrc` benötigt, da sie Informationen zum Bauen des Dokumentes mit Glossar und Akronymen enthält. Wird kein Glossar- und/oder Akronymverzeichnis benötigt, empfiehlt es sich, die entsprechende Option in der Klasse zu verwenden (siehe Kapitel 3). `.latexmkrc`

Beim Kompilieren des Dokumentes entsteht ein neuer Ordner `out`. Dieser Ordner enthält verschiedene Kompilierungsschritte, Log- und Synchronisierungsdateien von LaTeX. Im Zweifel kann dieser Ordner gelöscht und der Buildprozess neu gestartet werden. Die darin befindliche PDF-Datei (das gebaute Dokument) sollte jedoch vorher gespeichert werden, um einen Verlust der Daten zu vermeiden. Vorschläge für eine Konfiguration von LaTeX-Umgebungen in verschiedenen IDEs werden auch noch vervollständigt.

`out`

Zuletzt befindet sich in `hbrs-thesis.cls` sämtliche Konfiguration für die Klassendatei. Diese Konfiguration kann nach Belieben angepasst werden. Sollten Grundsätzliche Änderungs- oder Verbesserungsvorschläge an dieser Datei entstehen, bitte ich darum, diese auch bei GitHub (siehe `https://github.com/blackapple113/H-BRS-Thesisvorlage`) einzureichen.

`hbrs-thesis.cls`

3 Optionen der Klasse hbrs-thesis

Für die Modularität der Klasse wurden einige Optionen eingebaut. Mithilfe dieser Optionen lässt sich das Dokument zum Teil in Optik und Verwendungszweck anpassen. Im Folgenden werden die Optionen mit Beispielen erläutert.

3.1 Spracheinstellungen

Eine Pflichtoption, welche für eine korrekte Worttrennung verwendet werden muss ist die Einstellung der Sprache. Die Sprache kann aktuell nur zwischen englisch und deutsch unterschieden werden. Weitere Anpassungsmöglichkeiten für komplexere bilinguale Dokumente oder andere Spracheinstellungen müssen über die Datei `hbrs-thesis.cls` manuell eingestellt werden. Angenommen es wird ein Dokument auf Deutsch geschrieben, so muss in Kombination mit dieser Klasse die Option auf `german` gesetzt werden. Intern werden dann Optionen für die Deutsche Sprache gesetzt. Wird das Dokument auf Englisch geschrieben, so muss `english` angegeben werden.

```
\documentclass[german]{hbrs-thesis}
...
```

Werden bilinguale Spracheinstellungen im Dokument benötigt, müssen in der Klassendatei `hbrs-thesis.cls` Einstellungen für das Paket `babel` getroffen werden. Das Paket befindet sich unter der Region `Required packages`. Ist die Hauptsprache des Dokuments Englisch, kommen aber durchaus Deutsche Begriffe darin vor, eignet sich die Konfiguration `\RequirePackage[ngerman,english]{babel}`. Die zuletzt angegebene Sprache ist die Hauptsprache.

> **Information:** Wird die Sprache über die Optionsmöglichkeiten für das Paket `babel` direkt in `hbrs-template.cls` geändert, dürfen keine Sprachangaben als Klassenoptionen gesetzt werden.

3.2 Stiloptionen

Die Klasse besitzt ein paar Optionen, um verschiedene Stile anzubieten. Mithilfe der Option `classicstyle` bzw. `modernstyle` kann zwischen einer Schriftart mit Serifen und einer serifenlosen Schriftart (was sind Serifen: `https://de.wikipedia.org/wiki/Serife`) unterschieden werden. Serifen sollen dem Lesenden helfen die Zeilen besser zu verfolgen, um

während des Lesens nicht in der Zeile zu verrutschen, sind aber auf Bildschirmen teilweise schlechter darzustellen. Der Standard ist hier `modernstyle` und muss somit nicht explizit angegeben.

```
\documentclass[german,classicstyle]{hbrs-thesis}
...
```

Abbildung 3.1: Beispielbild für die Option `classicstyle`.

Neben dem Stil der Schriftart kann auch zwischen `report` und `article` unterschieden werden. Report bietet einzelne Kapitel beschrieben durch `\chapter{...}` welche immer auf einer neuen Seite anfangen. Im Gegensatz dazu bietet `article` lediglich Abschnitte beschrieben durch `\section{...}`, die nicht jedes Mal auf einer neuen Seite beginnen. Diese beiden Stile können durch die Größe des entstehenden Dokumentes unterschieden werden. Zum Beispiel wird so bei einem Praxisprojektbericht `article` verwendet wohingegen für eine Thesis `report` geeignet erscheint. Da die Dokumente nicht doppelseitig ausgedruckt werden entfällt die Option für ein Buchlayout. Diese wird vielleicht später noch hinzugefügt. **Die Option `article` ist standardmäßig konfiguriert.** Für eine Thesis sollten die Optionen für die Klasse `hbrs-thesis` also z. B. wie folgt verwendet werden:

```
\documentclass[german,classicstyle,report]{hbrs-thesis}
...
```

Für den Fall, dass der zweispaltige Stil wie in IEEE- oder ACM-Dokumenten optisch bevorzugt wird, existiert die Option `twocolumn`. Diese Eignet sich am besten in Kombination der Optionen `classicstyle` und `article`. Die entsprechende Konfiguration sieht dann beispielsweise folgendermaßen aus, wobei `article` Standard ist, also nicht angegeben werden muss:

```
\documentclass[german,classicstyle,article,twocolumn]{hbrs-thesis}
...
```

Abbildung 3.2: Beispielbild für die Option `twocolumn` in Kombination mit `article` und `classicstyle`.

3.3 Glossar und Akronymverzeichnis

Werden kein Glossar und/oder Akronymverzeichnis verwendet empfiehlt es sich die Option `noglossaries` in der Klasse zu verwenden. Damit wird das entsprechende Paket nicht geladen, es wird keine Warnung diesbezüglich ausgegeben und das Kompilieren geht eventuell schneller.

```
\documentclass[german,noglossaries]{hbrs-thesis}
...
```

4 Generelles

4.1 Referenzen

Die Funktion `\label{}` in LaTeXdient dazu, Markierungen innerhalb des Textes zu setzen, auf die später mithilfe von Referenzen zugegriffen werden kann. Dies ist besonders nützlich, um Verweise auf Abschnitte, Kapitel, Gleichungen, Abbildungen oder Tabellen innerhalb eines LaTeX-Dokuments zu erstellen. Label sollten für die einfachere Zuordnung aussagekräftig beschrieben sein. So bietet es sich zum Beispiel an für Kapitel (engl. chapter) `ch:` an den Anfang des Labels zu schreiben, für Abschnitte (engl. section) `sec:`, usw (vgl. Tabelle 4.1). Indem man ein Label mit einem eindeutigen Namen an der gewünschten Stelle platziert, kann man später im Text mithilfe von `\autoref{}` auf dieses Label verweisen, um automatisch die Bezeichnung inkl. Nummer einzufügen. Dies erleichtert die Aktualisierung von Referenzen, wenn sich die Nummerierung oder Anordnung im Dokument ändert, da LaTeX automatisch die korrekten Nummern aktualisiert.

Tabelle 4.1: Präfixe für Referenzen

Prefix	Bezeichnung
ch:	chapter
sec:	section
subsec:	subsection
fig:	figure
subfig:	sub figure
tab:	table
subtab:	sub table
eq:	equation
code:	code listing
itm:	enumerated list item
alg:	algorithm
app:	Anhang

```
\subsection{Beispiel für Referenzen}
\label{subsec:beispiel-fuer-referenzen}

Wie in \autoref{subsec:beispiel-fuer-referenzen} beschrieben...
```

4.1.1 Beispiel für Referenzen

Wie in Unterabschnitt 4.1.1 beschrieben...

Bei der Vergabe eindeutiger Bezeichner für die Referenzierung sollte darauf geachtet werden, dass lediglich ASCII-Zeichen verwendet werden. Je nach Kodierung der Quelldatei können Sonderzeichen sonst zu Fehlermeldungen führen. Sollten Teilelemente (sub figure, sub table) referenziert werden existieren zwei Möglichkeiten. Angenommen man benennt ein Teilelement mit `\label{subfig:testbild}`, kann mit `\autoref{sub@subfig:testbild}` allein auf die Nummerierung des Teilelements verwiesen werden, wohingegen mit `\autoref{subfig:testbild}` die Abbildungsnummer der übergeordneten Beschriftung vorangestellt wird. Beispiele und näheres dazu in Kapitel 5.

4.1.2 Weitere Referenzmöglichkeiten

`\label{}` Gibt dem zu referenzierenden Objekt einen Namen, auf den später referenziert werden kann.

`\ref{}` Referenziert auf das Objekt und gibt nur die Nummer für dieses Objekt zurück (sollte nur selten verwendet werden).

`\pageref{}` Gibt die Seitenzahl des referenzierten Objektes zurück.

`\autoref{}` Referenziert auf das Objekt mit der Bezeichnung und der Nummer. **Dies ist die präferierte Variante.**

`\nameref{}` Gibt den Inhalt des naheliegendsten relevanten Objekt zurück. So wird bei einer Überschrift diese zurückgegeben, aber für ein Bild der Inhalt der Bildunterschrift.

4.2 Fließumgebungen

Fließumgebungen (engl. floats) sind ein wichtiger Bestandteil von LaTeX-Dokumenten, um Grafiken, Abbildungen, Tabellen und ähnliche Inhalte flexibel und ästhetisch ansprechend im Layout zu platzieren. Durch die Verwendung von Fließumgebungen können diese Elemente automatisch an geeigneten Stellen im Text platziert werden, um den Lesefluss nicht zu unterbrechen. Innerhalb von Fließumgebungen kann der Befehl `\caption{}` verwendet werden, um Bild- oder Tabellenbeschriftungen hinzuzufügen, sowie der Befehl `\label` (siehe Abschnitt 4.1), um Labels für Referenzen festzulegen.

Trotz der automatischen Platzierung von Fließumgebungen in LaTeXgibt es Situationen, in denen eine manuelle Kontrolle über die Positionierung erforderlich ist. LaTeXbietet Optionen, um Fließumgebungen an bestimmten Stellen im Text zu platzieren. Die Option `[h]` (here) erlaubt eine Platzierung am Ort des Befehls im Text. `[t]` (top) platziert die Umgebung oben auf einer Seite, `[b]` (bottom) unten auf einer Seite und `[p]` (page) auf einer separaten Seite nur für Fließumgebungen. Soll die Fließumgebung fix an die vom Autor angegebene Stelle platziert werden, kann die Option `[H]` (Here!) verwendet werden. Diese Option zwingt LaTeXzur Positionierung an der angegebenen Stelle. Manuelle Positionierungsoptionen sollten jedoch sparsam verwendet werden, da sie die typografische Ästhetik beeinträchtigen könnten.

4.2.1 Beschriftungen

In LaTeXermöglicht der Befehl `\caption{}` das Hinzufügen von Beschriftungen zu Fließumgebungen. Diese Beschriftungen bieten eine kurze Erklärung oder Bezeichnung für das jeweilige Element. Beschriftungen sind hilfreich, um den Inhalt für den Leser zu erläutern und in den Kontext des Dokuments zu setzen. Darüber hinaus generiert LaTeXautomatisch die richtige Nummerierung für Abbildungen und Tabellen, was die Konsistenz und Verwaltung erleichtert. Durch die Kombination von `\caption{}` mit `\label{}` können Sie außerdem Referenzen zu diesen Elementen erstellen, die im Text automatisch die korrekte Nummer und

Bezeichnung anzeigen (vgl. Unterabschnitt 4.1.2). Mit dem Befehl `\captionof{float}{}` können Beschriftungen auch außerhalb von Fließumgebungen für bestimmte Elemente gesetzt werden.

Für Tabellen werden in wissenschaftlichen Artikeln die Beschriftungen häufig oberhalb des Elements gesetzt, da die Beschriftung unterhalb der Tabelle bei langen Tabellen zu spät gelesen wird. Bei Bildern hingegen sind die Beschriftungen unterhalb des Elements. Da die Formatierung von LaTeXdiesen Unterschied bei Tabellen nicht berücksichtigt existiert in diesem Dokument der Befehl `\captionabove{}`, welcher den Abstand zwischen Beschriftung und Tabelle korrigiert.

Werden mehrere Tabellen oder Bilder in einer Fließumgebung nebeneinander gezeigt sollten auch die Teilelemente einzelne Beschriftungen bekommen. Diese können mit dem Befehl `\subcaption{}` erstellt werden.

4.2.2 Zentrierung

Es ist oft sinnvoll, Fließumgebungen wie Tabellen oder Bilder zu zentrieren, insbesondere wenn sie nicht die volle Textbreite ausfüllen. Dies trägt zur ästhetischen Gestaltung des Dokuments bei und sorgt dafür, dass diese Elemente optisch ausgerichtet und präsentiert werden. Es gibt zwei gängige Möglichkeiten, dies in LaTeXzu erreichen.

Die erste Möglichkeit ist die Verwendung der `center`-Umgebung, welche **keine** Fließumgebung darstellt:

```
\begin{center}
    \includegraphics[width=0.4\columnwidth]
    ↳ {assets/images/bilder/pexels-pixabay-147411.jpg}
    \captionof{figure}{Zentrierung des Bildes mit \texttt{center}}
\end{center}
```

Abbildung 4.1: Zentrierung des Bildes mit `center`

Die zweite Möglichkeit ist die Verwendung des Befehls `\centering` innerhalb einer Fließumgebung:

```
\begin{figure}
    \centering
    \includegraphics[width=0.4\columnwidth]
    ↳ {assets/images/bilder/pexels-pixabay-147411.jpg}
    \caption{Einbetten des Bildes in eine Fließumgebung}
\end{figure}
```

Abbildung 4.2: Einbetten des Bildes in eine Fließumgebung

Beide Methoden zentrieren den Inhalt horizontal im Dokument. Mit der `center`-Umgebung erhalten wir jedoch keine Fließumgebung, weshalb wir für eine Beschriftung des Bildes den Befehl `\captionof{figure}{}` verwenden müssen. Wie in Abschnitt 4.2 beschrieben ist auch keine optimierte Positionierung des Elements durch LaTeXmöglich.

4.3 Minipage

Mit der Umgebung `minipage` können in LaTeXElemente wie Bilder, Tabellen aber auch Texte in ihrer Größe begrenzt und nebeneinander platziert werden. Eine `minipage`-Umgebung wird wie folgt definiert:

```
\begin{minipage}[outer position][height][inner pos]{width}
    % Inhalt
\end{minipage}
```

`width` Die Breite ist eine Pflichtangabe und beschreibt, wie breit die `minipage` sein soll. Hierbei empfehlen sich angaben wie bei Bildern mit `\columnwidth`, `\textwidth` oder `\linewidth`.

`outer position` Gibt an, wie die `minibox` ausgerichtet werden soll.

`height` Die Größe der `minibox` ist dynamisch an den Inhalt angepasst. Mit dieser Angabe kann die Höhe der Box verändert werden.

`inner pos` Gibt die vertikale Positionierung des Textes innerhalb der Box an, wenn die Box größer ist, als der Text.

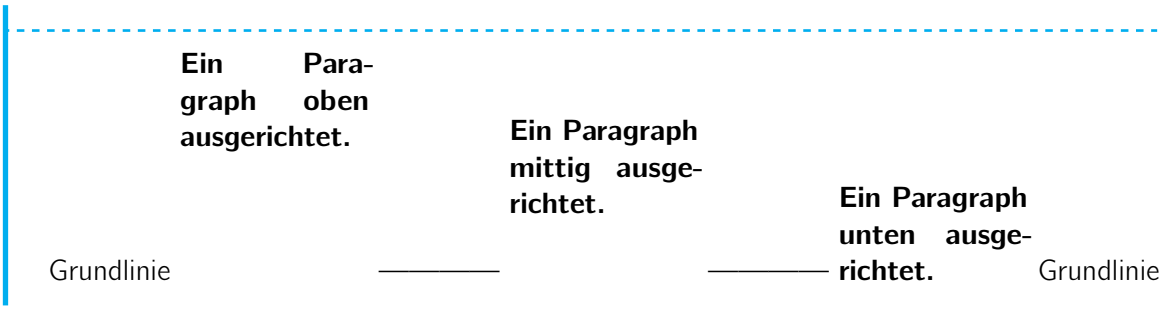
4.3.1 Äusere Positionierung (outer position)

```
Grundlinie
\begin{minipage}[b]{0.17\linewidth}
    \textbf{Eine \texttt{minipage} unten ausgerichtet.}
\end{minipage}
------------
\begin{minipage}{0.17\linewidth}
    \textbf{Eine \texttt{minipage} mittig ausgerichtet.}
\end{minipage}
------------
\begin{minipage}[t]{0.17\linewidth}
    \textbf{Eine \texttt{minipage} oben ausgerichtet.}
\end{minipage}
Grundlinie
```

Grundlinie **Eine** `minipage` **unten ausgerichtet.** **Eine** `minipage` **mittig ausgerichtet.** **Eine** `minipage` **oben ausgerichtet.** Grundlinie

4.3.2 Innere Positionierung (inner position)

```
Grundlinie
\begin{minipage}[b][3cm][t]{0.17\linewidth}
    \textbf{Ein Paragraph oben ausgerichtet.}
\end{minipage}
------------
\begin{minipage}[b][3cm][c]{0.17\linewidth}
    \textbf{Ein Paragraph mittig ausgerichtet.}
\end{minipage}
------------
\begin{minipage}[b][3cm][b]{0.17\linewidth}
    \textbf{Ein Paragraph unten ausgerichtet.}
\end{minipage}
Grundlinie
```


Ein Paragraph oben ausgerichtet.
Ein Paragraph mittig ausgerichtet.
Ein Paragraph unten ausgerichtet.
Grundlinie
Grundlinie

5 Bilder

Für qualitativ hochwertige Dokumente sollten möglichst hochauflösende Bilder oder SVG-Grafiken (`https://de.wikipedia.org/wiki/Scalable_Vector_Graphics`) verwendet werden, da besonders im Druck schlechte Auflösungen negativ hervorstechen.

Bilder können in die *Fließumgebung* `figure` gepackt werden. Für Positionierungsoptionen siehe Abschnitt 4.2. Das Bild innerhalb der Fließumgebung `figure` wird mit dem Befehl `\includegraphics[options]{path}` durch das Paket `graphicx` eingefügt. In `options` sollte immer die Option `width=\columnwidth` gesetzt werden. Bei einem einspaltigen Layout kann alternativ zu `\columnwidth` auch `\textwidth` verwendet werden. Diese Breitenangaben können auch durch Modifikatoren verändert werden. So kann das Bild zum Beispiel auf 70 % der Textbreite mit der Option `width=0.7\textwidth` skaliert werden.

```
\begin{figure}[Hhtbp] % Here!, here, top, bottom, page
    \centering % ← zentriert das Bild
    \includegraphics[width=0.7\columnwidth]
    ↳ {assets/images/bilder/pexels-pixabay-147411.jpg}
    \caption{Holzhaus am Gebirgssee}
    \label{fig:holzhaus-am-gebirgssee}
\end{figure}
```

Abbildung 5.1: Holzhaus am Gebirgssee

5.1 Vektorgrafiken (SVG)

Sollen komplexe Informationen auf klare und ansprechende Weise wie in Infografiken oder Diagrammen dargestellt werden, eignen sich Vektorgrafiken. Im Gegensatz zu rasterbasierten Formaten (wie JPEG oder PNG) behalten SVG-Grafiken ihre Qualität und Schärfe unabhängig von der Zoomstufe oder Größe. Um SVG-Grafiken in LaTeX-Dokumente einzufügen, gibt es verschiedene Ansätze, welche im folgenden kurz vorgestellt werden.

5.1.1 Konvertierung zu PDF oder EPS

Werden Grafiken zum Beispiel mit *draw.io*[1], *Inkscape*[2] oder anderen Werkzeugen selbst erstellt, empfiehlt es sich diese auch immer als PDF zu exportieren. Beim Export sollte darauf geachtet werden, dass die Größe des Bildes nur die gezeichneten Inhalte umfasst. Ich persönlich markiere dafür alle gezeichneten Elemente und exportiere nur die Auswahl, wobei es hier verschiedene Möglichkeiten gibt, da beim Markieren Teile des Bildes übersehen werden können. Die als PDF exportierten Bilder können wie normale Bilder mit dem Befehl `\includegraphics[options]{path/to/file.pdf}` in das Dokument eingebettet werden.

```
\begin{figure}
    \centering
    \includegraphics[width=0.5\columnwidth]
    ↳ {assets/images/bilder/landscape.pdf}
    \caption{Landschaft als PDF-Grafik}
    \label{fig:landschaft-als-pdf-grafik}
\end{figure}
```

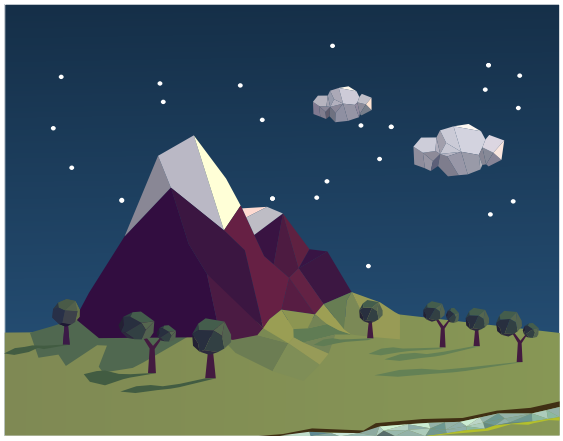


Abbildung 5.2: Landschaft als PDF-Grafik

[1] https://www.drawio.com/
[2] https://inkscape.org/de/

5.1.2 Direkte Verwendung von SVG-Grafiken

Bei der direkten Verwendung werden die SVG-Grafiken mit den Befehl `\includesvg[options]{path/to/file.svg}` zum Beispiel innerhalb eine Fließumgebung (siehe Abschnitt 4.2) geladen. **Durch das Paket wird Inkscape ausgeführt**, weswegen es notwendig ist dieses Programm auf dem Rechner installiert zu haben. Außerdem ist es notwendig für die Ausführung mit z. B. `latexmk` die Option `-shell-escape` hinzuzufügen. Weitere Informationen für die lokale Einrichtung des Computers zur Verwendung dieses Templates sind auf `https://github.com/blackapple113/H-BRS-Thesisvorlage`.

Setup Guide in README

```
\begin{figure}
    \centering
    \includesvg[width=0.5\columnwidth]{assets/images/bilder/landscape.
    ↪ svg}
    \caption{Landschaft als SVG-Grafik}
    \label{fig:landschaft-als-svg-grafik}
\end{figure}
```

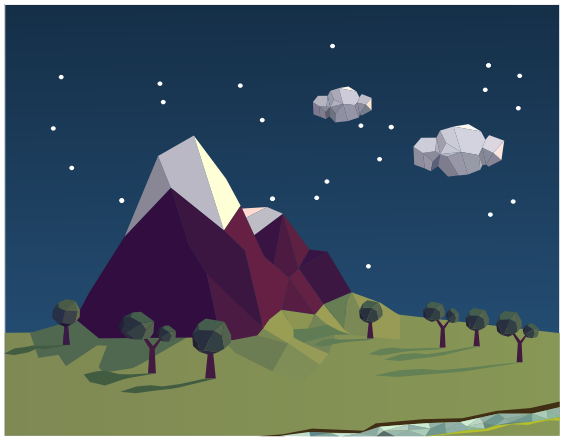
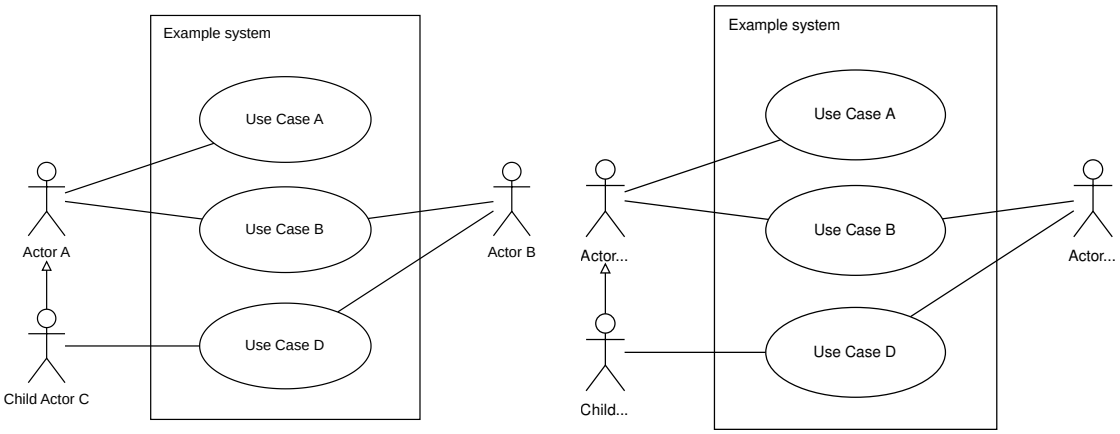


Abbildung 5.3: Landschaft als SVG-Grafik

5.1.3 Vergleich SVG- vs. PDF-Grafik

Werden SVG-Grafiken direkt in LaTeXverwendet werden sie in ein LaTeXverständliches Format umgewandelt. Dabei können ungewollte Formatierungsprobleme innerhalb der Grafik auftreten. Als Beispiel werden im Folgenden zwei Diagramme gezeigt. Das Diagramm auf der linken Seite wurde als PDF importiert und das Diagramm auf der rechten Seite als SVG-Grafik. Es sei gesagt, dass die klasse `hbrs-thesis` schon einige Optionen für die bessere Verwendung von SVG-Grafiken eingestellt hat. Dennoch kann es, wie im Beispiel gezeigt, zu Problemen kommen.



5.4.a: PDF-Grafik

5.4.b: SVG-Grafik

Abbildung 5.4: SVG- vs. PDF-Grafik

Die in diesem Beispiel gezeigten Unterschiede sind so gering, dass sie bei der ersten Kontrolle vielleicht gar nicht auffallen. Bei genauerer Betrachtung kann man erkennen, dass die Beschriftungen der einzelnen Aktoren in der SVG-Grafik nicht ausgeschrieben sind. Es fehlen also relevante Informationen, die in der PDF-Grafik enthalten sind. Trotzdem handelt es sich bei der PDF-Datei um eine Vektorgrafik, welche also keinen Qualitätsverlust aufweist.

5.2 Teilbilder

Wie in Unterabschnitt 5.1.3 als Vergleichsabbildung zwischen SVG- und PDF-Grafiken verwendet, müssen manchmal mehrere Bilder nebeneinander platziert werden. Diese können unabhängig voneinander Beschriftet werden oder als Teil der übergeordneten Fließumgebung. Um Bilder nebeneinander platzieren zu können, wird die Umgebung `minipage` (siehe Abschnitt 4.3) verwendet. Im Folgenden ein paar Beispiele für die Anordnung von Bildern in einer Reihe.

> **Information:** Die Nummerierung der Bilder in den folgenden Beispielen, sowie im Abbildungsverzeichnis ist etwas durcheinander. Das liegt an der Art und Weise, wie die Beispiele präsentiert werden. Unter Verwendung des dargestellten Codes passieren diese Fehler nicht.

```
\begin{figure}
    \centering
    \begin{minipage}[c]{0.49\columnwidth}
        \includegraphics[width=\linewidth]
        ↳ {assets/images/bilder/pexels-pixabay-147411.jpg}
        \subcaption{Holzhütte am Gebirgssee}
    \end{minipage}
    \hfill
    \begin{minipage}[c]{0.49\columnwidth}
        \includegraphics[width=\linewidth]
        ↳ {assets/images/bilder/pexels-david-bartus-1166209.jpg}
        \subcaption{Lavendelfelder im Sonnenuntergang}
    \end{minipage}
    \caption{Zwei farbenprächtige Naturfotografien}
\end{figure}
```

5.5.a: Holzhütte am Gebirgssee

5.5.b: Lavendelfelder im Sonnenuntergang

Abbildung 5.6: Zwei farbenprächtige Landschaftsfotos

```
\begin{figure}[H]
    \begin{minipage}[c]{0.27\columnwidth}
        \includegraphics[width=\linewidth,height=6cm,keepaspectratio]
        ↪ {assets/images/bilder/pexels-eberhard-grossgasteiger-2437291.
        ↪ jpg}
        \subcaption{Landschaft}
    \end{minipage}
    \hfill
    \begin{minipage}[c]{0.72\columnwidth}
        \includegraphics[width=\linewidth,height=6cm,keepaspectratio]
        ↪ {assets/images/bilder/pexels-pixabay-147411.jpg}
        \subcaption{Holzhütte am Gebirgssee}
    \end{minipage}
    \caption{Zwei farbenprächtige Landschaftsfotos}
\end{figure}
```

5.7.a: Landschaft

5.7.b: Holzhütte am Gebirgssee

Abbildung 5.8: Zwei farbenprächtige Landschaftsfotos

5.3 Wrapfigure

Die `wrapfigure`-Umgebung in LaTeXermöglicht es, Bilder in den Textfluss einzufügen und den umgebenden Text harmonisch um die Abbildung zu positionieren. Im Gegensatz zu herkömmlichen Fließumgebungen, die eine separate Zeile für Abbildungen reservieren, kann `wrapfigure` das Bild links oder rechts vom Text umfließen lassen. Dies ist besonders nützlich, wenn ein Bild eng mit dem umgebenden Text verbunden ist und eine nahtlose Integration gewünscht ist. Wie in einer normalen Fließumgebung können Beschriftungen mit `\caption{}` gesetzt werden. Die Verwendung von `wrapfigure`-Umgebung erfordert

eine sorgfältige Handhabung, da sie den Textfluss beeinflussen kann. Sie sollte mit Bedacht verwendet werden, um sicherzustellen, dass der Lesefluss und das Layout des Dokuments nicht gestört werden. Für weitere Informationen siehe `https://ctan.org/pkg/wrapfig` 2.

Tabelle 5.1: Optionen für `wrapfigure` und `wraptable`

Option		Platzierung
r	R	rechts, fließend rechts
l	L	links, fließend links
i	I	Innenseite, fließend Innenseite
o	O	Außenseite, fließend Außenseite

```
...

\begin{wrapfigure}{r}{0.5\columnwidth}
    \centering
    \includegraphics[width=0.5\columnwidth]
    ↪ {assets/images/bilder/pexels-pixabay-147411.jpg}
\end{wrapfigure}

...
```

Abbildung 5.9: Hütte am See vor dem Kursivgebirge

Weit hinten, hinter den Wortbergen, fern der Länder Vokalien und Konsonantien leben die Blindtexte. Abgeschieden wohnen sie in Buchstabhausen an der Küste des Semantik, eines großen Sprachozeans. Nicht einmal von der allmächtigen Interpunktion werden die Blindtexte beherrscht – ein geradezu unorthographisches Leben. Eines Tages aber beschloß eine kleine Zeile Blindtext, ihr Name war Lorem Ipsum, hinaus zu gehen in die weite Grammatik. Der große Oxmox riet ihr davon ab, da es dort wimmele von bösen Kommata, wilden Fragezeichen und hinterhältigen Semikoli, doch das Blindtextchen ließ sich nicht beirren. Es packte seine sieben Versalien, schob sich sein Initial in den Gürtel und machte sich auf den Weg. Als es die ersten Hügel des Kursivgebirges erklommen hatte, warf es einen letzten Blick zurück auf die Skyline seiner Heimatstadt Buchstabhausen, die Headline von Alphabetdorf und die Subline seiner eigenen Straße, der Zeilengasse. Wehmütig lief ihm eine rhetorische Frage über die Wange, dann setzte es seinen Weg fort.

6 Code

Diese Klasse verwendet das Paket `minted` (siehe `https://www.ctan.org/pkg/minted`) für Syntax Highlighting. Das Paket basiert auf *Pygments* (siehe `https://pygments.org/`) welches mit Python auf dem System installiert sein sollte, solange kein Docker-Container zum Bauen bereitsteht. Außerdem muss für das Kompilieren die Option `-shell-escape` hinzugefügt werden, also `latexmk -shell-escape file.tex`. Die von `minted` unterstützten sprachen für Highlighting sind unter folgender Adresse aufgelistet: `https://pygments.org/docs/lexers/`

6.1 Inline Code

Code, der im Text stehen soll, kann auf verschiedene Arten erreicht werden. Die einfachste ist die Verwendung von `\texttt{...}`. Dabei wird allerdings kein Syntax Highlighting verwendet. Der Text wird innerhalb dieser Umgebung nur an Leerzeichen umgebrochen und nicht innerhalb der Wörter.

Eine weitere Möglichkeit für Syntax Highlighting im Fließtext bietet `minted` mit dem Befehl `\mintinline{sprache}{codeblock}`. Dieser Befehl kann zusätzlich an vorgegebenen Stellen den Text umbrechen, was dennoch zu Problemen bei der Formatierung am Rand führen kann.

6.2 Block Code

Oft ist es notwendig Programmcode im Dokument darstellen zu können. Für diesen Zweck gibt es zwei verschiedene Möglichkeiten. Zum einen kann der Code direkt in die LaTeX-Datei geschrieben werden, was bei Änderungen zu manuellen Anpassungen führt. Zum anderen kann der Code auch aus den Dateien importiert werden. **Aufgrund von Limitierungen von minted und TeX kann es zur Ausführung von Code kommen, weshalb nur bekannter Code geladen werden sollte!**

Code kann wie in Kapitel 3 ohne starkes visuelles Absetzen direkt unter den Text geschrieben werden, in dem die Umgebung `codeblock` verwendet wird.

```
\begin{code}{python}
    if __name__ == "__main__":
        print("Hello World!")
\end{code}
```

```
if __name__ == "__main__":
    print("Hello World!")
```

Um den Code optisch durch Linien klar abzugrenzen, wird die Umgebung `codeblock` verwendet. Dieser Codeblock eignet sich vor allem für größere Codebereiche. Zusätzlich können weitere von `\NewTCBInputListing` bereitgestellte Optionen konfiguriert werden (siehe `https://www.ctan.org/pkg/tcolorbox`).

```
\begin{codeblock}{python}{Hello World in Python}[label=
↳ {code:python-class}]
    if __name__ == "__main__":
        print("Hello World!")
\end{codeblock}
```

Code 6.1: Hello World in Python

```
1 if __name__ == "__main__":
2     print("Hello World!")
```

Wie oben schon beschrieben kann Code auch aus anderen Dateien geladen werden. Dies geschieht mithilfe des Befehls `\inputcode{language}{path/to/file}{title}`.

```
\inputcode{java}{assets/code/Test.java}{Java Testdatei}
```

Code 6.2: Java Testdatei

```
1 public class Test{
2     public static void main(String[] args){
3         System.out.println("Hello World!");
4     }
5 }
```

Soll nur ein bestimmter Bereich an Zeilennummern aus Code-Dateien angezeigt werden können die Zeilen als Zeilennummern (jeweils inklusiv) angegeben werden. Generell können über diesen Shortcut weitere von `\NewTCBInputListing` bereitgestellte Optionen konfiguriert werden (siehe `https://www.ctan.org/pkg/tcolorbox`).

```
\inputcode[minted options={firstline=2,lastline=4}]{java}
↳ {assets/code/Test.java}{Java Testdatei Zeilen 2 - 4}
```

Code 6.3: Java Testdatei Zeilen 2 - 4

```
2 public static void main(String[] args){
3     System.out.println("Hello World!");
4 }
```