

# Die powerdot-Dokumentation \*

Hendri Adriaens      Christopher Ellison<sup>†</sup>

v1.4g (2011/04/25)

## Zusammenfassung

Powerdot ist eine Präsentationsklasse für  $\LaTeX$ , welche die schnelle und einfache Entwicklung von professionellen Präsentationen erlaubt. Sie enthält viele Möglichkeiten, die Präsentation zu verbessern und den Vortragenden zu unterstützen, beispielsweise automatische Overlays, das Einfügen persönlicher Anmerkungen und einen Handout-Modus. DVI, PS und PDF sind mögliche Ausgabeformate, um die Präsentation anzuzeigen. Ein leistungsstarkes Template-System erlaubt das einfache Entwickeln neuer Stile. Eine  $\LaTeX$ -Layoutdatei ist enthalten.

## Inhalt

<b>1</b>	<b>Einführung</b>	<b>2</b>	<b>8</b>	<b>Kompilieren der Präsentation</b>	<b>28</b>
			8.1	Abhängigkeiten . . . . .	28
<b>2</b>	<b>Das Erstellen einer Präsentation</b>	<b>3</b>	8.2	Erzeugen und Ansehen der	
	2.1 documentclass-Optionen . . .	3		Output-Datei . . . . .	28
	2.2 Aufbau-Optionen . . . . .	6	<b>9</b>	<b>Einen eigenen Folienstil erstellen</b>	<b>29</b>
<b>3</b>	<b>Das Erstellen von Folien</b>	<b>10</b>	9.1	Generelle Informationen . . . . .	29
	3.1 Die Titelfolie . . . . .	10	9.2	Paletten definieren . . . . .	30
	3.2 Andere Folien . . . . .	11	9.3	Definition von Vorlagen . . . . .	31
<b>4</b>	<b>Overlays</b>	<b>12</b>	9.4	Überwachung des Setup . . . . .	31
	4.1 Der <code>\pause</code> -Befehl . . . . .	12	9.5	Hauptkomponenten . . . . .	32
	4.2 Listenumgebungen . . . . .	13	9.6	Das Folieninhaltsverzeichnis . .	34
	4.3 Der <code>\item</code> -Befehl . . . . .	14	9.7	Sonstige Optionen . . . . .	36
	4.4 The <code>\onslide</code> -Befehl . . . . .	15	9.8	Voreingestellte Templates . . . .	37
	4.5 Relative Overlays . . . . .	16	9.9	Der Hintergrund . . . . .	37
<b>5</b>	<b>Präsentationsstrukturen</b>	<b>16</b>	9.10	Titelfolie, Titel und Abschnitte .	37
	5.1 Kapitel herstellen . . . . .	16	9.11	Das Testen des Stils . . . . .	38
	5.2 Das Erstellen einer Übersicht . .	17	<b>10</b>	<b>Die Benutzung von <math>\LaTeX</math> für Präsentationen</b>	<b>38</b>
<b>6</b>	<b>Sonstiges</b>	<b>18</b>	10.1	Wie das Layout benutzt wird . .	39
	6.1 Anmerkungen . . . . .	18	10.2	Unterstützung der Syntax . . . .	40
	6.2 Leere Folien . . . . .	19	10.3	Programmübersetzung mit $\LaTeX$ .	41
	6.3 Die Bibliographiefolie . . . . .	19	10.4	Erweiterung des Layouts . . . . .	41
	6.4 Wortwörtliche Wiedergabe auf		<b>11</b>	<b>Fragen</b>	<b>42</b>
	Folien . . . . .	20	11.1	Häufig gestellte Fragen . . . . .	42
	6.5 Der <code>\twocolumn</code> -Befehl . . . . .	21	11.2	Mailingliste . . . . .	44
<b>7</b>	<b>Zur Verfügung stehende Stile</b>	<b>23</b>	<b>12</b>	<b>Quelltextdokumentation</b>	<b>44</b>
			<b>13</b>	<b>Implementierung</b>	<b>45</b>
			13.1	Allgemeiner Aufbau . . . . .	45

\*Diese Dokumentklasse kann heruntergeladen werden von den CTAN Spiegelservern: `/macros/latex/contrib/powerdot`. Sehen Sie in `powerdot.tex`, wenn Sie Installationsinformationen für das Einfügen von powerdot in ihre  $\LaTeX$ -Distribution und zur Lizenz dieser Dokumentklasse benötigen. Diese Dokumentation wurde angepasst von Herbert Voß.

<sup>†</sup>Die Übersetzung in die deutsche Sprache wurde von Lennart Petersen, Christiane Matthieu, Christoph Köhler, Julia Babigian und Christine Römer (Institut für Germanistische Sprachwissenschaft der Friedrich-Schiller-Universität Jena) 2009 vorgenommen, Die 2. Übersetzungsversion erstellte im November 2011 Christine Römer.

# 1 Einführung

Mithilfe dieser Klasse ist es möglich, professionell aussehende Folien herzustellen. Die Klasse wurde so entworfen, dass die Entwicklung von Präsentationen möglichst einfach gehalten ist, sodass man sich auf den eigentlichen Inhalt der Präsentation konzentrieren kann, anstatt sich mit technischen Details befassen zu müssen. Nichtsdestoweniger sind grundlegende  $\text{\LaTeX}$ -Kenntnisse für die Nutzung von Powerdot notwendig.

Diese Klasse baut auf die prosper-Klasse [9] und das HA-prosper-Paket [1] auf und erweitert diese. Das HA-prosper-Paket war anfangs dafür gedacht, prosper zu erweitern und einige Fehler und Probleme dieser Klasse zu korrigieren. Beim Voranschreiten der Entwicklung des HA-prosper-Paketes bemerkte man, dass nicht alle Probleme auf diese Weise gelöst werden konnten. Diese Entdeckung war der Beginn eines neuen Projektes, aus dem eine neue Klasse hervorgehen sollte, welche die sonst nötige Kombination von prosper und HA-prosper ersetzen konnte. Die Powerdot-Klasse ist das Ergebnis dieses Projektes.

Der verbleibende Abschnitt dieses Kapitels sei nun einem Blick auf die Struktur einer Powerdotpräsentation und einem Überblick über die gesamte Dokumentation gewidmet.

Der Aufbau einer Präsentation ist stets derselbe. Er ist im folgenden Beispiel nachvollziehbar.

```
\documentclass[<class options>]{powerdot}
\pdsetup{<presentation options>}
\begin{document}
  \begin{slide}{a slide}
    Contents of the slide.
  \end{slide}
  \section{first section}
  \begin{slide}[<slide options>]{another slide}
    Contents of the slide.
  \end{slide}
  \begin{note}{personal note}
    The note.
  \end{note}
\end{document}
```

Einige dieser Elemente definieren die Struktur des Dokuments. Zuvorderst sind einige Optionen zum `\documentclass`-Befehl möglich, welche die Art der Ausgabe des Dokuments betreffen, beispielsweise das Papierformat. Diese Optionen werden in Abschnitt 2.1 besprochen. Ferner gibt es Spezifikationsoptionen, die einige globale Elemente der Präsentation wie zum Beispiel Fußnoten kontrollieren. Sie werden in Abschnitt 2.2 behandelt.

Sind diese Entscheidungen getroffen, können die slide-Umgebung zum Erstellen von Folien (siehe Abschnitt 3) und die note-Umgebung zum Erstellen von Anmerkungen, die mit den Folien angezeigt werden (siehe Abschnitt 6.1), genutzt werden. Overlays ermöglichen, Inhalte nacheinander anzuzeigen, was in Abschnitt 4 beschrieben wird. Der `\section`-Befehl dient der Strukturierung der Präsentation. Dieses wird in Abschnitt 5 dargestellt. Abschnitt 7 zeigt eine Übersicht über die Stile, die in dieser Klasse enthalten sind und deren Charakteristiken. Abschnitt 8 behandelt die Möglichkeiten der Ausgabe und beinhaltet damit wichtige Informationen bezüglich der für Powerdot benötigten Pakete.

Abschnitt 9 ist am interessantesten für jene, die ihren eigenen Stil für powerdot entwickeln oder einen bestehenden modifizieren wollen. Abschnitt 10 erklärt, wie  $\text{\LaTeX}$  [6] genutzt werden kann, um Powerdotpräsentationen zu erstellen. Das Ende der Dokumentation schließlich (Abschnitt 11) wurde einem Kapitel über Fragen wie „Wo kann ich Beispiele finden?“ gewidmet und enthält ferner die Informationen, wohin man sich wenden kann, wenn die eigenen Fragen zu Powerdot noch nicht gelöst sein sollten.

## 2 Das Erstellen einer Präsentation

Dieser Abschnitt beschreibt die möglichen Optionen zur Kontrolle der Ausgabe der Präsentationen und dem Erscheinungsbild eben jener.

### 2.1 documentclass-Optionen

Begonnen wird mit den Optionen, die in den `\documentclass`-Befehl mittels einer durch Kommata separierten Liste eingefügt werden. Für jede Möglichkeit wird der Standardwert<sup>1</sup> in der Beschreibung erwähnt. Das ist der Wert, der genutzt wird, wenn entschieden wurde, einer Option keinen Wert zu geben oder die Option gar nicht zu verwenden.

*option* Diese Optionen bestimmen die Art der Ausgabe, die produziert werden soll. Der  
*mode* Standardwert ist `present`.

`mode=present`

Dieser Modus wird dazu genutzt, die eigentliche Präsentation zu erstellen. Er aktiviert Overlays und Übergangseffekte. Auf Overlays wird in Abschnitt 4 näher eingegangen.

`mode=print`

Dieser Modus kann verwendet werden, um die Folien inklusive ihrer visuellen Aufmachung, aber ohne Overlays oder Übergangseffekte auszudrucken.

`mode=handout`

Dieser Modus erstellt in Schwarzweiß einen Überblick über die Folien, der genutzt werden kann, um persönliche Notizen darauf zu verzeichnen, die Präsentation an Studenten auszugeben, sie als eigenen Leitfaden einzusetzen, etc.

*option*  
*nohandoutpagebreaks*

`nohandoutpagebreaks`

Standardmäßig gibt der `handout`-Modus Dokumente mit zwei Folien pro Seite aus. Wenn mehr Folien pro Seite platziert werden sollen, füge man diese Option in den `\documentclass`-Befehl ein und `powerdot` wird es  $\LaTeX$  überlassen zu entscheiden, wann der Seitenumbruch stattfinden soll, meistens also, wenn eine Seite gefüllt ist.

*option*  
*nohandoutframes*

`nohandoutframes`

Im `handout`-Modus ist jede Folie standardmäßig in einem Rahmen. Diese Option stellt diese Rahmen aus.

*option*  
*paper*

Die Option `paper` hat drei mögliche Werte. Der Standardwert ist `screen`.

`paper=screen`

Dies ist ein Spezialformat mit optimalem Bildschirmverhältnis (4/3). Die entsprechenden Seitenlängen sind 8.25 Inch zu 11 Inch (ungefähr 21cm zu 28cm). Dieses Papierformat ist für den `print`- oder `handout`-Modus nicht vorhanden. In diesen Modi wird `powerdot` automatisch zur Größe DIN A4 wechseln und eine Warnung diesbezüglich in die Logdatei der Präsentation schreiben.

`paper=a4paper`

Das Format DIN A4 wird für die Präsentation oder das Handout genutzt.

<sup>1</sup>Das ist der Wert, der als Standard genutzt wird, wenn nichts anderes explizit gewählt wird.

`paper=letterpaper`

Das amerikanische letter-Papierformat wird genutzt.

`paper=smartboard`

„smartboard size“ (Großbild) wird genutzt. (`papersize={900pt,1440pt}`).

Einige wichtige Informationen zum Papierformat, zum Kompilieren und dem Betrachten von Präsentationen finden sich im Abschnitt 8.

*option orient* Die Option `orient` kontrolliert die Ausrichtung der Präsentation. Der Standardwert ist `landscape` (Querformat).

`orient=landscape`

Die Präsentation erhält das Querformat. Dieser Wert ist im `handout`-Modus nicht verfügbar. Sollte er dennoch gewählt werden, wird ihn `powerdot` zum Hochformat verändern und einen entsprechend warnenden Vermerk in die Logdatei schreiben.

`orient=portrait`

Dies erstellt Folien im Hochformat. Es ist zu beachten, dass nicht alle Stile diese Ausrichtung unterstützen. In Abschnitt 7 sind diesbezüglich entsprechende Informationen für jeden Stil zu finden.

*option display* Die Option `display` beeinflusst die Produktion von Folien und Anmerkungen. Der Standardwert ist `slides`.

`display=slides`

Dies wird nur die Folien in der Präsentation anzeigen.

`display=slidesnotes`

Dies wird sowohl die Folien als auch die Anmerkungen in der Präsentation anzeigen. Mehr Informationen bezüglich der Anmerkungen enthält Abschnitt 6.1.

`display=notes`

Dies wird nur die Anmerkungen anzeigen.

Hier nun noch einige weitere Optionen, welche die Ausgabe beeinflussen.

*option size*

`size`

Dies ist die Schriftgröße von normalem Text in Punkten (points). Mögliche Werte sind 8pt, 9pt, 10pt, 11pt, 12pt, 14pt, 17pt und 20pt. Der Standardwert ist 11pt.<sup>2</sup>

*option style*

`style`

Dies definiert den Stil, der für die Präsentation geladen werden soll. Standardmäßig wird der `default`-Stil geladen. Mögliche Varianten sind im Abschnitt 7 zu finden.

*option fleqn*

`fleqn`

Diese Option setzt die Gleichungen linksbündig. Eben so, wie es die `gleichnamige` Option in der `article`-Klasse ebenfalls setzt.

*option leqno*

`leqno`

Nimmt Gleichungsnummern rechts. Die gleichen wie bei der `article`-Klasse.

---

<sup>2</sup>Es sollte beachtet werden, dass von 10pt, 11pt und 12pt abweichende Schriftgrößen keine Standardvarianten sind und man für diese das `extsizes`-Paket [11], welches diese Größen beinhaltet, installieren muss.

*option*  
nopsheader

#### nopsheader

Standardmäßig schreibt powerdot einen Postscriptbefehl in die PS-Datei, um sicherzugehen, dass die nachfolgenden Bearbeitungsprogramme, wie ps2pdf, wissen, welches Papierformat sie nutzen sollen, auch wenn keine explizite Spezifikation dazu in der Kommandozeile steht. Dazu ist Abschnitt 8 zu beachten. Wenn Probleme mit den nachfolgenden Bearbeitungsprogrammen oder dem Ausdrucken auftreten, oder wenn man selbst das Papierformat in den nachfolgenden Bearbeitungsstufen definieren möchte, nutzt man diese Option.

*option*  
hlentries

#### hlentries

Dies hebt Einträge des Inhaltsverzeichnisses hervor, wenn der Eintrag mit der aktuellen Folie übereinstimmt. Der Standardwert ist `true`. Man beachte dazu auch Abschnitt 5. Wenn keine solchen Hervorhebungen gewünscht werden (beispielsweise im `print`-Modus), nutzt man `hlentries=false`.

*option*  
hlsections

#### hlsections

Dies hebt die Abschnitte (sections) des Inhaltsverzeichnisses hervor, wenn diese mit den Abschnitten der laufenden Präsentation übereinstimmen. Der Standardwert ist `false`. Dazu ist ebenfalls Abschnitt 5 zu beachten. Eine Spezifikation dieser Option wendet die Hervorhebung der Abschnitte an. Dies kann hilfreich sein, wenn man einen Stil nutzt, der ein geteiltes Inhaltsverzeichnis enthält.

*option*  
pauseslide

#### pauseslide

Diese Option fügt eine leere Folie (normalerweise eine schwarze) auf der ersten Seite der Präsentation ein und wird ferner automatisch zur zweiten Seite wechseln, wenn die Präsentation in einem PDF-Viewer wie dem Acrobat Reader geöffnet wird. Ferner fügt diese Option einen Verweis bei jeder Folien- oder Abschnittsüberschrift ein, der zurück auf die leere Folie führt. Klickt man dann dagegen irgendwo auf die Pause-Folie, kommt man zurück zu jener Folie, auf der man sich zuvor befand. Diese Option kann dazu genutzt werden, bei einer Präsentation temporär zu pausieren, wenn man beispielsweise einen Beweis an der Tafel anführen möchte. Sie können auch eine andere Farbe als schwarz wählen, beispielsweise `pauseslide=white`.

*options*  
clock

#### clock

Dies zeigt eine kleine Digitaluhr auf den Folien, die dafür genutzt werden kann, die Zeit während des Vortrages im Auge zu behalten.

Hier ist ein Beispiel für einen `\documentclass`-Befehl.

```
\documentclass[
  size=12pt,
  paper=screen,
  mode=present,
  display=slidesnotes,
  style=tycja,
  nohandoutpagebreaks,
  pauseslide,
  fleqn
]{powerdot}
```

Dieses Beispiel definiert eine Präsentation im `tycja`-Stil, mit schwarzer Folie, einer Schriftgröße von 12 Punkten und linksbündigen Gleichungen.

```
\documentclass[
  size=12pt,
```

```

paper=letterpaper,
mode=handout,
display=slidesnotes,
style=tycja,
nohandoutpagebreaks,
pauseslide,
fleqn
]{powerdot}

```

Mit dem Wechsel der `paper`- und `mode`-Optionen wird nun ein Handout mit mehr als zwei Folien pro Seite, ganz so, wie es die `nohandoutpagebreaks`-option definiert, ausgegeben.

## 2.2 Aufbau-Optionen

`\pdsetup` Es gibt einige Zusatzoptionen, die dabei helfen, eine Präsentation den eigenen Wünschen anzupassen. Diese Optionen sind allerdings nicht im `\documentclass`-Befehl enthalten, was technisch begründet ist.<sup>3</sup> Wir unterscheiden zwei Arten von Optionen. Jene Optionen, die nur global, also die gesamte Präsentation betreffen, mittels des `\pdsetup`-Befehls definiert werden können und jene Optionen, die sowohl global (mittels `\pdsetup`) als auch lokal (mittels Folienumgebungen, mehr dazu in Abschnitt 3) nutzbar sind.

### 2.2.1 Globale Optionen

Dieser Abschnitt beschreibt Optionen, die einzig global in der Präambel der Präsentation mittels des `\pdsetup`-Befehls genutzt werden können.

*option*  
palette

**palette**

Dies definiert die nutzbare Farbpalette. Eine Farbpalette ist eine Sammlung von Farben, die von einem Stil bereitgestellt wird. Abschnitt 7 erläutert genau, welche Stile welche Farbpaletten enthalten.

*option*  
theslide

**theslide**

Diese Option legt fest, wie die Foliennummern auf der Folie gesetzt werden. Der Standardwert ist `\arabic\{slide}\~/\pageref*{lastslide}`, was in einer Form wie 5/22 dargestellt wird. Man beachte, dass der Befehlszeilenabschnitt `\{arabic{slide}` die Nummer der gerade angezeigten Folie und der Befehlszeilenabschnitt `\pageref*{lastslide}` die Nummer der letzten Folie anzeigt.<sup>4</sup>

*option*  
thenote

**thenote**

Dies ist gleich dem `theslide`-Befehl, bezieht sich allerdings auf die Foliennummern der Anmerkungen. Der Standardwert ist `note~\arabic{note}~of~slide~\arabic{slide}` und `\arabic{note}`, wobei hier die Nummer der aktuellen Anmerkung auf der aktuellen Folie gezeigt wird. Dies könnte beispielsweise so aussehen: `note 2 of slide 7`.

*option*  
counters

**counters**

Die `counters`-Option listet Zähler auf, die möglicherweise bei Overlays geschützt werden sollen; da Material von Overlays (sehen Sie dazu Kapitel 4) mehrfach verarbeitet wird, also auch  $\LaTeX$ -Zähler wie `equation`. Deshalb ist es möglich, dass zu hohe Zählungen entstehen. Um zu vermeiden, dass die Zähler bei

<sup>3</sup>Der interessierte Leser wende sich dazu an den Abschnitt bezüglich des `xkvtxp`-Paketes in der `xkeyval`-Dokumentation [2].

<sup>4</sup>Es wird `\pageref` mit Stern verwendet, was mittels `hyperref` mit `hyperref` definiert wird und so wird kein Verweis auf die Seite, auf die sich bezogen wird, erstellt.

verschiedenen Overlays unterschiedliche Nummern anzeigen, nutzt man diese Option. Die `equation-`, `table-`, `figure-`, `footnote-` und `mpfootnote-`Zähler sind bereits geschützt. Wenn man weitere Zähler benutzt, beispielsweise für Theoreme, sollte man diese mittels dieser Option auflisten. Zum Beispiel:

```
counters={theorem,lemma}
```

option  
list

**list**

Diese Option enthält eine ganze Liste von Optionen, die dem `enumitem`-Paket angepasst sind, welches das Layout jener Listen, die mithilfe der `enumerate` und `itemize`-Umgebungen erzeugt werden, definiert. Ein Beispiel:

```
list={labelsep=1em,leftmargin=*,itemsep=0pt,topsep=5pt,parsep=0pt}
```

Man beachte für mehr Informationen bezüglich des Layouts von Listen das `enumitem`-Paket [4].

options  
enumerate  
itemize

**enumerate**   **itemize**

So wie die `list`-Option, aber nur `enumerate`- und `itemize`-Umgebungen kontrollierend.

option  
clockformat

**clockformat**

Diese Option spezifiziert das Format der Uhr. Das Format wird mit der `Acrobatutil.printd`-Funktion gesetzt.<sup>5</sup> Der Standardwert ist `HH:MM:ss`, der zeigt eine „00-23 hour, 00-59 minute, 00-59 second clock“. Ein Beispiel:

```
clockformat=h:MM tt
```

Die obigen Einstellungen zeigen an der Uhr 1-12 Stunden, 00-59 Minuten, am/pm-Uhrzeit. Dies heißt, die Uhr könnte 5:53 Uhr pm (p.m. „nach dem Mittag“) anzeigen.

option  
clockrefresh

**clockrefresh**

Diese Option aktualisiert die Uhr über eine Zahl, die angibt, wie oft in Millisekunden aktualisiert werden soll. Das Standardverhalten aktualisiert die Uhr jede Sekunde. Der Defaultwert ist 1000. Beachten Sie, wenn das `clockformat` ohne Sekundeneinstellung ist, dann macht es keinen Sinn, die Uhr jede Sekunde zu aktualisieren. Ein entsprechendes Beispiel:

```
clockrefresh=60000
```

Dies bedeutet, dass die Uhr jede Minute aktualisiert wird.

### 2.2.2 Globale und lokale Optionen

Dieser Abschnitt beschreibt Optionen, die sowohl global mittels des `\pdsetup`-Befehls als auch lokal mittels Folienumgebungen (mehr dazu in Abschnitt 3) definiert werden können.

options  
lf  
cf  
rf

**lf**   **cf**   **rf**

Diese bestimmen den Inhalt der Fußzeile nach links, mittig oder rechts, die leer voreingestellt ist.

option  
trans

**trans**

<sup>5</sup>Eine komplette Liste der möglichen Formate findet man in *Acrobat JavaScript Scripting Reference*[13].

Diese Option definiert, dass der Standardübergangseffekt für die Präsentation genutzt wird. Diese Übergangseffekte werden erst nach der Kompilierung der Präsentation ins PDF-Format angezeigt. Man beachte dazu auch Abschnitt 8. Die folgenden Übergangseffekte werden unterstützt: Split, Blinds, Box, Wipe, Dissolve, Glitter und Replace. Wenn mit einem PDF-Viewer gearbeitet wird, der PDF 1.5 anzeigen kann, sind ebenfalls Fly, Push, Cover, Uncover oder Fade nutzbar. Es ist wichtig zu beachten, dass die meisten PDF-Viewer technisch empfindlich sind, beispielsweise wird box nicht funktionieren.

Der Standardeffekt ist Replace, welcher einzig eine Folie durch die nächste ersetzt, wenn sie angesehen werden. Man beachte, dass manche PDF-Viewer (wie der Acrobat Reader 5 oder höher) die Übergangseffekte nur im Vollbildmodus anzeigen. Wenn ein bisher nicht aufgeführter Übergangseffekt (beispielsweise ein Wischeffekt mit einer speziellen Bewegungsrichtung) genutzt werden soll, ist das durchaus möglich. powerdot wird eine Warnung in die Logdatei schreiben, dass der ausgewählte Effekt möglicherweise nicht vom PDF-Viewer angezeigt werden wird. Hier ein Beispiel, das funktioniert:

```
trans=Wipe /Di 0
```

Der Acrobat Reader zeigt diesen Wischeffekt von links nach rechts, statt, wie es der Standard ist, von oben nach unten. Für weitere Informationen sei das PDF-Referenzhandbuch empfohlen.

option  
method

method

Diese Option kann dann genutzt werden, wenn die Präsentation spezielles Material enthält, das nicht in der „üblichen Art“ von L<sup>A</sup>T<sub>E</sub>X behandelt werden soll. Verbatim-Material kann beispielhaft angeführt werden. Mögliche Werte sind direct und file. Diese Optionen werden genauer im Abschnitt 6.4 erklärt.

options  
logohook  
logopos  
logocmd

logohook   logopos   logocmd

Wenn logopos spezifiziert wurde, wird ein Logo vom logocmd-Wert definiert und auf jede Folie gesetzt. Die Position des Logos kann in relativen Werten abhängig von der Breite und Höhe der Folien gesetzt werden. {0,0} ist die untere linke Ecke des Papiers und {\slidewidth,\slideheight} ist die obere rechte Ecke. Für die Positionierung des Logos wird der \rput-Befehl von pstricks [16, 17] genutzt. Dieser Befehl erlaubt es ebenfalls, einen genauen Punkt als Logoposition zu definieren. Dieser Punkt kann mithilfe der logohook-Option eingetragen werden und die Werte t1, t, tr, r, Br, br, b, b1, B1, l, B und c sind möglich. Weitere Informationen bezüglich des \rput-Befehls enthält die pstricks-Dokumentation. Hier ist ein Beispiel, in dem die Blume des default-Stiles in den husky-Stil integriert wird.

```
\documentclass[style=husky]{powerdot}
\pdsetup{
  logohook=t,
  logopos={.088\slidewidth,.99\slideheight},
  logocmd={\includegraphics[height=.08\slideheight]{powerdot-default.ps}}
}
\begin{document}
...
\end{document}
```

Der Standardwert von logohook ist t1.

Eine besondere Eigenschaft von powerdot, mit deren Hilfe Präsentationen ein wenig Leben eingehaucht werden kann, ist die Nutzung von Zufallspunkten. Diese Punkte werden irgendwo auf den Folien gesetzt und nutzen die Farben der ausgewählten

Farbpalette. Auch Overlays werden dieselben Punkte aufweisen. Diese Eigenschaft basiert auf `random.tex` [3]. Verschiedenste Optionen sind möglich, um das Erscheinen der Zufallspunkte zu kontrollieren.

*option*  
randomdots `randomdots`

Standardmäßig sind die Zufallspunkte ausgeschaltet. Sie werden dann generiert, wenn diese Option mit dem Wert `true` versehen wird, während der Wert `false` sie wiederum ausschaltet. Wenn kein expliziter Wert zu dieser Option gesetzt ist, wird `true` angenommen.

*options*  
dmindots `dmindots` `dmaxdots`

Die Anzahl der Punkte pro Folie ist ebenfalls zufällig. Diese Optionen legen die minimale und maximale Anzahl Punkte pro Folie fest. Die Standardwerte sind 5 bzw. 40.

*options*  
dminsize `dminsize` `dmaxsize`

Dies ist der minimale und maximale Radius der Punkte. Standardwerte sind 5pt bzw. 40pt.

*options*  
dminwidth `dminwidth` `dmaxwidth` `dminheight` `dmaxheight`

Diese Optionen bestimmen das Areal auf der Folie, in dem die Zufallspunkte erscheinen sollen. Der Standardwert dieser Option definiert, dass die Punkte überall auf der Folie erscheinen können, aber das ist variabel, beispielsweise so, dass die Punkte nur noch im Textfeld vorkommen. Die Standardwerte sind 0pt, `\slidewidth`, 0pt, `\slideheight`.

Hier ein Beispiel, das Punkte in einem kleineren Rechteck auf der Folie erlaubt.

```
\pdsetup{
  dminwidth=.1\slidewidth,dmaxwidth=.9\slidewidth,
  dminheight=.2\slideheight,dmaxheight=.8\slideheight
}
```

*option*  
dbright `dbright`

Diese Option dient der Helligkeitseinstellung der Punkte. Der Wert sollte eine Zahl zwischen -100 und 100 sein. Wenn die Zahl negativ ist, wird die Farbe zu Schwarz abgedunkelt, wobei -100 Schwarz ergibt. Ist die Zahl dagegen positiv, wird die Farbe zu Weiß aufgehellt, wobei wiederum 100 Weiß ergibt. Ist der Folienhintergrund hell, nutzt man eher einen positiven Wert der Option `bright`, ist er dagegen dunkel, sollte man sich wohl für einen negativen Wert entscheiden. Der Standardwert ist 60, was eine Mischung aus 40% der Originalfarbe und 60% Weiß ergibt.

*option*  
dprop `dprop`

Mithilfe dieser Option können zusätzliche Parameter zum `\psdot`-Befehl hinzugefügt werden, welcher die Zufallspunkte erstellt. Beispielsweise der Stil oder die Linienstärke der Punkte sind so veränderbar. Für weitere Informationen bezüglich des `/psdot`-Befehls sei auf die `pstricks`-Dokumentation [16, 17] verwiesen. `powerdot` enthält zwei zusätzliche Stile, die für Zufallspunkte eingesetzt werden können. Diese sind `ocircle` (offener Kreis) und `osquare` (offenes Quadrat).

Hier sind zwei Beispiele für den Gebrauch von Zufallspunkten.

```
\pdsetup{
  randomdots,dminwidth=.2\slidewidth
}
```

Dieses Beispiel beinhaltet Zufallspunkte, verhindert aber, dass die linken 20% der Folie von ihnen genutzt wird.

```
\pdsetup{
  randomdots,dprop={dotstyle=ocircle,linewidth=.5pt},
  dminsize=500pt,dmaxsize=600pt,dmindots=2,dmaxdots=5
}
```

Dieses Beispiel setzt höchstens fünf große Kreise auf die Folie. Diese Kreise sind zu groß, um vollständig auf der Folie angezeigt werden zu können, weswegen man nur Teile von ihnen als Kurven sehen wird.

### 2.2.3 \pdsetup-Beispiel

Hier ein Beispiel für einen pdsetup-Befehl, mit dem man eine Präsentation beginnen könnte.

```
\pdsetup{
  lf=My first presentation,
  rf=For some conference,
  trans=Wipe,
  theslide=\arabic{slide},
  randomdots,dmaxdots=80
}
```

Dies setzt die linke und rechte Fußzeile und initialisiert den Wipe-Übergangseffekt. Ferner beinhaltet die Foliennummerierung nicht die Nummer der letzten Folie, sondern einzig jene der aktuellen. Und schließlich werden die Folien mit bis zu 80 Zufallspunkten bedeckt sein.

Hier ist eine kurze Anmerkung bezüglich des Erscheinens von Fußzeilen nötig. Die Foliennummer (definiert von der theslide-Option) wird in einer Fußzeile angezeigt werden. Die meisten Stile setzen sie in die rechte Fußzeile. Wenn Fußzeile und Foliennummer nicht leer sind, wird ~--~ zwischen sie eingefügt werden, um sie voneinander abzutrennen. Einige Stile modifizieren womöglich diese Standardvorgehensweise.

## 3 Das Erstellen von Folien

### 3.1 Die Titelfolie

```
\title
\author
\and
\date
\maketitle
```

Die Titelfolie wird mittels des \maketitle-Befehls erstellt.

```
\maketitle[<options>]
```

Dieser Befehl nutzt dieselben Werte wie das L<sup>A</sup>T<sub>E</sub>X-Standarddokument. Das optionale Argument <options> kann jede Option des Abschnittes 2.2.2 enthalten. Das Einfügen einer solchen Option in den \maketitle-Befehl betrifft nur die Titelfolie und keine andere. Man beachte das untere Beispiel.

```
\documentclass{powerdot}
\title{Title}
\author{You \and me}
\date{October 7, 2010}
\begin{document}
  \maketitle
  ...
\end{document}
```

Die `author-`, `title-` und `date-`Angaben definieren den beim Erstellen der Titelfolie zu nutzenden Text. Die Gestaltung der Titelseite wird durch den ausgesuchten Stil bestimmt. Man beachte den Gebrauch des `\and`-Befehls, um mehrere Autoren aufzuführen. Weitere Informationen bezüglich der Befehle wie `\title` und `\author` sind im  $\TeX$ -Handbuch [12] enthalten. .

### 3.2 Andere Folien

`slide` Das Herzstück jeder Präsentation sind die Folien. Bei Powerdot wird der Inhalt jeder Folie innerhalb einer `slide`-Umgebung definiert.

```
\begin{slide} [options] {<slide title>}
<body>
\end{slide}
```

Im Abschnitt 4 werden wir sehen, wie man die Folien etwas lebendiger gestalten kann. Bleiben wir jetzt aber erst einmal bei einem simplen Beispiel.

```
\begin{slide}{First slide}
Hello World.
\end{slide}
```

Die Folienumgebung hat ein notwendiges Argument, nämlich den Folientitel. Sobald eine Folie erstellt wird, wird der Folientitel dafür genutzt, einen Eintrag ins Inhaltsverzeichnis und in die Lesezeichenliste einzufügen. Das Inhaltsverzeichnis ist eine Liste der Folien- und Abschnittitel der Präsentation, die auf jeder Folie erscheint.

Die aufgeführten Titel des Inhaltsverzeichnisses sind mit deren Folien und Abschnitten verlinkt (sofern die Präsentation ins PDF-Format kompiliert wurde) und bieten somit eine praktische Hilfe, um innerhalb der Präsentation rasch zwischen benötigten Folien zu wechseln. Die Lesezeichenliste erscheint erst, wenn die Kompilierung ins PDF-Format abgeschlossen ist. Sie dient ebenfalls als eine Art Inhaltsverzeichnis, erscheint allerdings auf *keiner* der Folien, sondern in einem zusätzlichen Fenster im PDF-Viewer. Im obigen Beispiel werden die Einträge im Inhaltsverzeichnis wie in der Lesezeichenliste beide als `First slide` geführt.

Die `options` für die `slide`-Umgebung können jene Optionen enthalten, die im Abschnitt 2.2.2 aufgeführt werden. Zusätzlich können folgende Optionen benutzt werden.

`option`  
`toc`

`toc`

Wenn definiert, wird dieser Wert für den Eintrag ins Inhaltsverzeichnis genutzt; ansonsten wird dazu der Folientitel herangezogen. Wenn `toc=` definiert ist, wird kein Eintrag erstellt.

`option`  
`bm`

`bm`

Falls definiert, wird dieser Wert für den Eintrag in die Lesezeichenliste genutzt; ansonsten wird dazu der Folientitel herangezogen. Wenn `bm=` definiert ist, wird kein Eintrag erstellt.

Diese optionalen Argumente sind besonders nützlich, wenn der Folientitel sehr lang ist oder wenn er  $\TeX$ -Befehle enthält, die in den Lesezeichen nicht korrekt angezeigt werden würden.<sup>6</sup> Beim Erstellen der Einträge sollte darauf geachtet werden, spezielle Zeichen wie `'`, `,` und `=` zwischen den geschweiften Klammern `{` and `}` zu verstecken. Werfen wir einen Blick auf das Beispiel, das diese optionalen Argumente nutzt.

<sup>6</sup>Der Prozess, der die Lesezeichen setzt, nutzt `\pdfstringdef` vom `hyperref`-Paket und kann mit akzentuierten Zeichen wie `\i`.

```
\begin{slide}[toc=,bm={LaTeX, i*i=-1}]{\color{red}\LaTeX, $i^2=-1$}
  My slide contents.
\end{slide}
```

In diesem Beispiel wird der Folientitel als  $\text{\color{red}\LaTeX, } i^2 = -1$  erscheinen. Dieser Text wird nicht korrekt als Lesezeichen angezeigt werden. Es wurde also ein Versuch unternommen, dieses zu korrigieren, aber oftmals ergibt dies nicht denselben Text. Der genannte Titel würde als Lesezeichen folgendermaßen angezeigt werden: `redLaTeX, i2=-1`. Auf der anderen Seite wird der manuell erstellte Lesezeicheneintrag so angezeigt: `LaTeX, i*i=-1`. Man beachte, dass kein Eintrag im Inhaltsverzeichnis vorgenommen werden wird, da der Wert `toc=`.

Zusätzlich zur `slide`-Umgebung kann jeder individuelle Stil seine eigenen Umgebungen definieren. Viele Stile haben eine `wideslide`-Umgebung. Dahinter steht die Idee, dass gewisser Inhalt aus Platzgründen schlecht oder überhaupt nicht mit dem Inhaltsverzeichnis zusammen auf einer Folie realisiert werden kann. In diesem Fall ist es von Vorteil, eine Folie zu nutzen, die das Inhaltsverzeichnis nicht aufführt. Die `wideslide`-Umgebung enthält diese Funktion und bietet so mehr Platz für den eigentlichen Folieninhalt. Abschnitt 7 enthält mehr Informationen bezüglich der unterschiedlichen Umgebungen der einzelnen Stile.

## 4 Overlays

Oft möchte man nicht, dass alle Informationen auf einer Folie gleichzeitig erscheinen, sondern vielmehr, dass eine nach der anderen auftaucht. Bei `powerdot` wird das mittels `Overlays` realisiert. Jede einzelne Folie kann viele `Overlays` enthalten, wobei die `Overlays` eines nach dem anderen ausgegeben werden.

### 4.1 Der `\pause`-Befehl

`\pause` Der `\pause`-Befehl ist die einfachste Möglichkeit, Informationen aufeinanderfolgend auszugeben.

```
\pause [number]
```

Hier ein simples Beispiel:

```
\begin{slide}{Simple overlay}
  power\pause dot
\end{slide}
```

Der Informationstext auf einer Folie wird nur bis zum Befehl `\pause` ausgegeben, es erscheint also nichts anderes auf der Folie als dieses bestimmte Stück Text, solange nicht ein Klick mit der Maus erfolgt oder eine Taste gedrückt wird. Erst dann wird der weitere Inhalt der Folie ausgegeben, entweder bis diese keine weiteren Informationen mehr enthält oder bis zum nächsten `\pause`-Befehl innerhalb derselben Folie. In diesem Beispiel erscheint `power` mit dem ersten und `powerdot` mit dem zweiten `Overlay`. Der `\pause`-Befehl wird oft innerhalb von `itemize`- und `enumerate`-Umgebungen gebraucht, zum Beispiel:

```
\begin{slide}{Multiple pauses}
  power\pause dot \pause
  \begin{itemize}
    \item Let me pause\ldots \pause
    \item \ldots while I talk \pause and chew bubble gum. \pause
    \item Perhaps you'll be persuaded.
    \item Perhaps not.
  \end{itemize}
\end{slide}
```

Indem `\pause` vor der `itemize`-Umgebung verwendet wurde, erscheint kein Stichpunkt vor dem dritten Overlay. Danach wird ein Stichpunkt nach dem anderen ausgegeben, wobei jeder sein eigenes Overlay hat. Mehr Informationen bezüglich der Verwendung von Listen folgen im nächsten Abschnitt.

Ein optionales Argument (in eckigen Klammern) des `\pause`-Befehls spezifiziert die Nummer der Overlays, die als Pause fungieren. Ein Verwendungsbeispiel ist:

```
\begin{slide}{Pause longer}
\begin{itemize}
\item A \pause
\item B \pause[2]
\item C
\end{itemize}
\end{slide}
```

Bei diesem Beispiel erscheint Stichpunkt C mit dem vierten Overlay. Die Nützlichkeit dieser Möglichkeit wird im nächsten Abschnitt deutlicher, entsprechend werden wir dann noch mal ein ähnliches Beispiel betrachten.

## 4.2 Listenumgebungen

Die Listenumgebungen `itemize` und `enumerate` werden bei `powerdot` in besonderer Weise behandelt. Sie haben ein optionales Argument, das im `enumitem`-Paket enthalten ist (siehe [4]). `powerdot` liefert eine extra Verschlüsselung für dieses optionale Argument. In den folgenden Beispielen wird die `itemize`-Umgebung zur Auflistung der einzelnen Punkte verwendet, mit der `enumerate`-Umgebung funktioniert es aber genauso.

Hier ein Beispiel für den üblichen Gebrauch der `itemize`-Umgebung:

```
\begin{slide}{Basic itemize}
\begin{itemize}
\item A \pause
\item B \pause
\item C
\end{itemize}
\end{slide}
```

Die Ausgabe erfolgt, indem einfach mit jedem Overlay ein Stichpunkt nach dem anderen erscheint.

*option*  
type

Angenommen, wir wollten, dass alle Stichpunkte einer Folie zeitgleich auftauchen, dabei aber nur einer davon zum entsprechenden Zeitpunkt 'aktiv' sein soll. Das wird realisiert mittels der `type`-Option für die `itemize`-Umgebung. Der vorgegebene Wert ist 0.

```
\begin{slide}{Type 1 itemize}
\begin{itemize}[type=1]
\item A \pause
\item B \pause
\item C
\end{itemize}
\end{slide}
```

Jetzt wird jeder Stichpunkt in der *inaktiven Farbe* (die durch den verwendeten `powerdot`-Stil festgelegt ist) ausgegeben. Mit dem Overlay, mit dem ein Stichpunkt normalerweise erst erscheinen würde, bekommt dieser seine eigentliche Farbe und wird darüber aktiv. Das Standardverhalten ist mit `type=0` gegeben. Es ist immer noch möglich, optionale Argumente in `enumitem` über das zweite optionale Argument zu geben.

```
\begin{slide}{Type 1 itemize}
\begin{enumerate}[type=1][label=\romani*]
\item A \pause
\item B \pause
\end{enumerate}
\end{slide}
```

```

\item C
\end{enumerate}
\end{slide}

```

Achten Sie darauf, dass dies immer das zweite optionale Argument ist! Es ist möglich, das erste leer zu lassen, dies ist nur für powerdot gültig.

Listen können auch ineinander gestapelt sein, um so kompliziertere Strukturen zu schaffen. Wenn eine Liste eingebettet ist in eine andere, enthält sie die `type`-Option der ‘Mutter’-Liste. Das kann aber aufgehoben werden, indem `type`-Option beim optionalen Argument der eingebetteten Liste genauer definiert wird. Unser Beispiel zeigt nur ein mögliches, mittels gestapelter Listen produziertes Konstrukt, es können jedoch auch Konstrukte anderer Art und auf andere Weise gestaltet werden.

```

\begin{slide}{Nested lists}
\begin{itemize}
\item A\pause
\begin{itemize}[type=1]
\item B\pause
\end{itemize}
\item C
\end{itemize}
\end{slide}

```

Hier werden A und B mit dem ersten Overlay ausgegeben, aber B ist inaktiv. Erst mit dem zweiten Overlay wird B aktiv, mit Overlay 3 wird C sichtbar.

### 4.3 Der `\item`-Befehl

`\item` Der `\item`-Befehl hat bei powerdot noch ein extra Argument (*optional*), das eine flexiblere Produktion von Overlays erlaubt als der `\pause`-Befehl.

```

\item[⟨label⟩]⟨overlays⟩

```

Mit diesem optionalen Argument kann man spezifizieren, mit welchem Overlay ein bestimmter Stichpunkt ausgegeben wird. Diese Spezifikation ist eine durch ein Komma separierte Liste, wo jeder Stichpunkt die in Tabelle 1 angegebenen Notationen nutzen kann. Das `⟨label⟩`-Argument ist für das optionale Argument des `\item`-Befehls in

Syntax	Bedeutung
x	Nur Overlay x
-x	Alle Overlays bis zu und eingeschlossen x
x-	Alle Overlays von x, eingeschlossen x
x-y	Alle Overlays von x zu y, eingeschlossen x und y

Tabelle 1: `\item` und `\onslide`-Notation

$\LaTeX$ . Das  $\LaTeX$ -Handbuch [12] enthält noch mehr Informationen bezüglich dieses Arguments.

Hier ein Beispiel:

```

\begin{slide}{Active itemize}
\begin{itemize}[type=1]
\item<1> A
\item<2> B
\item<3> C
\end{itemize}
\end{slide}

```

Wie oben besprochen, sollte A nur bei der Überblendung 1, B nur bei Overlay 2 und C nur bei Overlay 3 aktiv sein, im inaktiven Status sollten die jeweiligen Stichpunkte wegen `type=1` in der inaktiven Farbe erscheinen.

Wenn die `type=0`-Option definiert ist und wenn jedem Stichpunkt eine Überblendungsoption gegeben wurde, erscheint jeder Stichpunkt nur, wenn er aktiv ist. Ist er inaktiv, wird er nicht auf der Folie angezeigt. Weitere Beispiele, die die Syntax für `<overlays>` demonstrieren, werden im nächsten Abschnitt diskutiert.

#### 4.4 The `\onslide`-Befehl

`\onslide` Overlays können auch unter Verwendung des `\onslide`-Befehls geschaffen werden.

```
\onslide{<overlays>}{<text>}
```

Der Befehl benötigt eine `<overlays>`-Spezifizierung als erstes Argument und den `<text>`, der auf der Folie erscheinen soll, als zweites Argument. Die `<overlays>`, auf denen der Text erscheinen wird, werden genauer definiert als eine durch ein Komma separierte Aufzählung mit der in Tabelle 1 dargestellten Syntax. Wir beginnen mit einem einfachen Beispiel:

```
\begin{slide}{Simple onslide}
  \onslide{1,2}{power}\onslide{2}{dot}
\end{slide}
```

Wir haben eingerichtet, dass `power` mit den Overlays 1 und 2 erscheint, `dot` nur mit Overlay 2. Wie bereits vermutet wird bei diesem Beispiel das Gleiche ausgegeben wie bei dem ersten `\pause`-Beispiel. Trotzdem ist es ein klarer Fall, dass ihre Syntax komplizierter ist. Diese Slight „Komplikation“ gestattet auch mehr Flexibilität.

`\onslide+` Betrachten wir das gleiche Beispiel mit den folgenden Modifikationen:

```
\begin{slide}{Simple onslide+}
  \texttt{onslide }: \onslide{1}{power}\onslide{2}{dot}
  \texttt{onslide+}: \onslide+{1}{power}\onslide+{2}{dot}
\end{slide}
```

Der `\onslide+`-Befehl gibt seinen Inhalt in völlig anderer Art und Weise aus. Jetzt erscheint `dot` mit jedem Overlay, allerdings wird es außer bei Overlay 2 *nur* in seiner inaktiven Farbe ausgegeben. Das ist vergleichbar mit dem `type=1`-Verhalten für die Listen (siehe Abschnitt 4.2).

Wenn wir dieses Beispiel ausführen, werden wir zudem feststellen, dass der `\onslide`-Befehl zunächst Material verbirgt, und doch die richtige Menge an Platz dafür reserviert. Bei Overlay 2 erscheinen die dots alle übereinander. Der nächste Befehl reserviert keinen Platz.

`\onslide*` Statt bestimmtes Material zu verbergen und Platz dafür zu reservieren (`\onslide`) oder `<text>` außer beim entsprechenden Overlay (`<overlays>`) in der inaktiven Farbe auszugeben (`\onslide+`), gibt dieser Befehl das Material einfach ohne jede weitere Formatierung aus. Betrachten wir folgendes Beispiel, um den Unterschied zu verstehen:

```
\begin{slide}{Simple onslide*}
  \texttt{onslide }: \onslide{1}{power}\onslide{2}{dot}
  \texttt{onslide+}: \onslide+{1}{power}\onslide+{2}{dot}
  \texttt{onslide*}: \onslide*{1}{power}\onslide*{2}{dot}
\end{slide}
```

Wir sind bereits vertraut mit der Ausgabe der ersten zwei Zeilen. Die dritte Zeile gibt `power` mit Overlay 1 und `dot` mit Overlay 2 aus, allerdings ist bei Overlay 2 kein Platz für `power` reserviert. Stattdessen wird `dot` an der gleichen Position des Cursors beginnen, bei der `power` mit dem ersten Overlay ausgegeben wurde, und es ist auch nicht in einer Linie unter den anderen dots angeordnet.

Abschließend betrachten wir ein Syntax-Beispiel, das sowohl mit `\item` als auch mit `\onslide` möglich ist. Noch mal zur Erinnerung: Diese Befehle benötigen eine durch ein Komma separierte Aufzählung für die genauere Definition des `<overlay>`-Arguments. Dabei kann jedes Element die in Tabelle 1 beschriebene Syntax nutzen. Die verschiedenen Varianten sind in folgendem Beispiel demonstriert:

```

\begin{slide}{Lists}
  \onslide{10}{on overlay 10 only}\par
  \onslide{-5}{on every overlay before and including overlay 5}\par
  \onslide{5}{on every overlay after and including overlay 5}\par
  \onslide{2-5}{on overlays 2 through 5, inclusive}\par
  \onslide{-3,5-7,9-}{on every overlay except overlays 4 and 8}
\end{slide}

```

## 4.5 Relative Overlays

Manchmal ist es sehr lästig, im Auge zu behalten, wann ein Stichpunkt auftauchen beziehungsweise aktiv werden soll, zum Beispiel, wenn man möchte, dass ein bestimmter Text auf dem entsprechenden Overlay *nach* einem speziellen Stichpunkt erscheinen soll. Abhilfe dafür leisten relative Overlays, die allerdings nicht außerhalb von `\item`-Listenumgebungen verwendet werden sollten. Betrachten wir ein einfaches, einleuchtendes Beispiel:

```

\begin{slide}{Relative overlays}
  \begin{itemize}
    \item A \pause
    \item B \onslide{+1}{(visible 1 overlay after B)}\pause
    \item C \onslide{+2}{(appears 2 overlays after C, visible until the end)}
    \pause
    \item D \onslide{+1-6}{(appears 1 overlay after D, visible until overlay 6)}
    \pause
    \item E \pause
    \item F \pause
    \item G \onslide{+1-+3}{(appears 1 overlay after G for 3 overlays)}\pause
    \item H \pause
    \item I \pause
    \item J \pause
    \item K
  \end{itemize}
\end{slide}

```

Wie zu sehen ist, wird auch hier der `\onslide`-Befehl genutzt, die einzige Veränderung der Syntax ist die Auflistung der Overlays. Dadurch kann ein '+'-Zeichen in der Liste genauer definiert werden. In der simpelsten Verwendung wird durch den `\onslide{+1}`-Befehl der entsprechende Text ein Overlay nach demjenigen ausgegeben, auf dem er *eigentlich* erschienen wäre. Nach wie vor kann die in Tabelle 1 dargestellte Syntax verwendet werden, demonstriert im oben stehenden Beispiel. `\onslide{+1-6}` bewirkt ebenfalls, dass der entsprechende Text ein Overlay nach demjenigen ausgegeben wird, auf dem er eigentlich erschienen wäre, und dass schon ausgegebene Textpassagen bis Overlay 7 gezeigt bleiben. In der letzten Demonstration des oben stehenden Beispiels wird gezeigt, wie man den Text einer ganzen Reihe von relativen Overlays erscheinen lassen kann.

## 5 Präsentationsstrukturen

### 5.1 Kapitel herstellen

`\section` Dieser Abschnitt beschreibt den `\section`-Befehl, der die Möglichkeit eröffnet, eine Präsentation zu strukturieren.

```

\section[<options>]{<section title>}

```

Dieser Befehl produziert eine Folie mit dem Abschnittstitel `<section title>` und nutzt außerdem den eingesetzten Text für die Repräsentation des entsprechenden Abschnitts

in einem Inhaltsverzeichnis und der Lesezeichenliste. Es gibt Möglichkeiten (*options*), um die Ausgabe zu kontrollieren.

*option*  
*tocsection* Diese Option kontrolliert die Repräsentation eines Abschnittes im Inhaltsverzeichnis. Der vorgegebene Wert ist `true`.

```
tocsection=true
```

So wird ein Abschnitt im Inhaltsverzeichnis geschaffen. Das bedeutet, dass alle nun folgenden Folien unter diesem Gliederungspunkt erscheinen, bis ein neuer eingefügt wird.

```
tocsection=false
```

Auf diese Art und Weise wird kein Abschnitt im Inhaltsverzeichnis geschaffen, somit wird dieser als normale Folie eingeordnet und aufgeführt.

```
tocsection=hidden
```

So wird ein Abschnitt im Inhaltsverzeichnis hergestellt, aber er ist nur sichtbar, wenn man eine Folie ansieht, die zu diesem Abschnitt gehört. Diese Funktion könnte verwendet werden, um einen zu diskutierenden Abschnitt an die Präsentation anzuhängen, der aber nur dann gebraucht und entsprechend gezeigt wird, wenn noch genügend Zeit für diese Diskussion ist.

*option*  
*slide* Diese Option bestimmt, ob der `\section`-Befehl eine Folie schafft. Der vorgegebene Wert ist `true`.

```
slide=true
```

Eine Folie wird erzeugt.

```
slide=false
```

Es wird keine Folie hergestellt. Wenn auch `tocsection false` hat, bewirkt der `\section`-Befehl gar nichts. Wenn ein Inhaltsverzeichnisabschnitt hergestellt wird (`tocsection= true` oder `hidden`), der Abschnitt aber selbst keine eigene Folie hat, verweist seine Verknüpfung auf die erste Folie unter diesem Abschnitt.

*option*  
*template* Diese Option kann verwendet werden, um eine Abschnittsfolie mit einer anderen Schablone zu erstellen. Im Defaultfall wird eine normale `slide`-Umgebung genutzt, um eine Abschnittsfolie herzustellen, aber wenn ein Stil andere Schablonen bietet, die für einen bestimmten Zweck genutzt werden könnten (zum Beispiel die `wideslide`-Umgebung), dann ermöglicht diese Option die Nutzung der entsprechenden Schablone. Abschnitt 7 gibt einen Überblick über die verschiedenen Stile und deren verfügbaren Schablonen.

Letztendlich können alle für normale Folien verfügbare Optionen auch für Folien verwendet werden, die mit dem `\section`-Befehl erstellt worden sind (siehe Abschnitt 3). Wenn ein Abschnitt mit der `tocsection`-Option erstellt wird, entfernen `toc=` oder `bm=` das einleitende Inhaltsverzeichnis oder das entsprechende Lesezeichen nicht.

## 5.2 Das Erstellen einer Übersicht

`\tableofcontents` Dieser Befehl erstellt eine Übersicht (Gliederung) für Präsentationen und kann nur für Folien genutzt werden.

```
\tableofcontents[options]
```

Es gibt diverse Befehle (*options*), um die Ausgabe dieses Befehls zu kontrollieren.

*option*  
*type* Diese Option bestimmt, ob gewisses Material (abhängig von der Eingabe für die `content`-Option weiter unten) versteckt oder in der inaktiven Farbe ausgegeben wird. Der vorgegebene Wert ist 0. Sie ist vergleichbar mit der `type`-Option für Listenumgebungen (Abschnitt 4.2).

`type=0`

Wenn Material nicht dem gefragten Typ (bei der `content`-Option spezifiziert) entspricht, wird es verborgen.

`type=1`

Wie `type=0`, nur dass das Material nicht versteckt, sondern in der inaktiven Farbe ausgegeben wird.

*option content* Die `content`-Option kontrolliert, welche Elemente in den Überblick aufgenommen werden. Der vorgegebene Wert ist `all`. Die unten stehende Beschreibung setzt die `type=0` voraus, es ist aber kein Problem, den alternativen Text für `type=1` daraus zu folgern.

`content=all`

So wird der vollständige Überblick einer Präsentation ausgegeben, einschließlich aller Abschnitte und der Folien, die nicht in Abschnitten versteckt sind (siehe Abschnitt 5.1).

`content=sections`

So werden nur die Abschnitte in der Präsentation ausgegeben.

`content=currentsection`

So wird nur der aktuelle Abschnitt ausgegeben.

`content=future`

So wird der gesamte Inhalt, beginnend bei der aktuellen Folie, ausgegeben.

`content=futuresections`

So werden alle Abschnitte, beginnend beim aktuellen Abschnitt, ausgegeben.

Diesen Abschnitt beendet ein kleines Beispiel, das demonstriert, wie man eine Präsentation entwickelt, die einen allgemeinen Überblick der Abschnitte in der Präsentation enthält, eine grundsätzliche Idee des Inhalts und für jeden Abschnitt eine detaillierte Übersicht über seine einzelnen Folien liefert.

```
\begin{slide}[toc,bm=]{Overview}
  \tableofcontents[content=sections]
\end{slide}
\section{First section}
\begin{slide}[toc,bm=]{Overview of the first section}
  \tableofcontents[content=currentsection,type=1]
\end{slide}
\begin{slide}{Some slide}
\end{slide}
\section{Second section}
...
```

## 6 Sonstiges

### 6.1 Anmerkungen

*note* Die `note`-Umgebung kann verwendet werden, um persönliche Anmerkungen in den eigentlichen Folientext einzugliedern. Die Ausgabe der Anmerkungen kann mittels der `display`-Option kontrolliert werden (siehe Abschnitt 2.1). Hier ist ein Beispiel:

```

\begin{slide}{Chewing gum}
...
\end{slide}
\begin{note}{Reminder for chewing gum}
  Don't forget to mention that chewing gum is sticky.
\end{note}

```

## 6.2 Leere Folien

`emptyslide` Die `emptyslide` Umgebung stellt eine völlig leere Folie her. Die Textbox auf der Folie könnte für spezielle Dinge verwendet werden, wie zum Beispiel die Ausgabe von Fotos. Diese Funktion erlaubt also auch das Erstellen und Wiedergeben einer Diashow. Zum Beispiel:

```

\begin{emptyslide}{}
\centering
\vspace{\stretch{1}}
\includegraphics[height=0.8\slideheight]{me_chewing_gum.eps}
\vspace{\stretch{1}}
\end{emptyslide}

```

Der `\includegraphics`-Befehl wird durch das `graphicx`-Paket [5] definiert. Der `\stretch`-Befehl wird verwendet, um das Bild vertikal zu zentrieren. Beide Befehle sind in Ihrem Lieblings- $\text{\LaTeX}$ -Handbuch beschrieben, zum Beispiel [12]. Man kann zudem die Längen `\slideheight` und `\slidewidth` nutzen, um das Bild maßstabsgetreu der Folie anzupassen.

## 6.3 Die Bibliografieolie

`thebibliography` `powerdot` benennt die standardisierte `article-thebibliography`-Umgebung um, damit die Erstellung einer Abschnittsüberschrift und fortlaufender Kopfzeilen unterdrückt wird. Alle anderen Bestandteile wurden beibehalten. Jeweils eines der folgenden Beispiele können Sie verwenden (abhängig davon, ob Sie  $\text{Bi}\text{\TeX}$  nutzen oder nicht):

```

\begin{slide}{Slide}
\cite{someone}
\end{slide}
\begin{slide}{References}
\begin{thebibliography}{1}
\bibitem{someone} Article of someone.
\end{thebibliography}
\end{slide}

```

```

\begin{slide}{Slide}
\cite{someone}
\end{slide}
\begin{slide}{References}
\bibliographystyle{plain}
\bibliography{YourBib}
\end{slide}

```

Für den Fall längerer Bibliographien, die auf mehrere Folien verteilt werden sollen, empfiehlt sich die Verwendung der Pakete `natbib` und `bibentry` [8]. Das erlaubt Folgendes:

```

\begin{slide}{References (1)}
\bibliographystyle{plain}
\nobibliography{YourBib}
\bibentry{someone1}
\bibentry{someone2}
\end{slide}
\begin{slide}{References (2)}
\bibentry{someone3}
\end{slide}

```

Werfen Sie einen Blick in Ihr Lieblings- $\text{\LaTeX}$ -Handbuch für weitere Informationen zum Zitieren und zu Bibliographien.

## 6.4 Wortwörtliche Wiedergabe auf Folien

*option*  
verbatim

powerdot hat drei verschiedene Methoden, Folien aufzubereiten, von denen zwei hauptsächlich entwickelt wurden, um das Einbeziehen wortwörtlichen Inhalts<sup>7</sup> auf Folien einfacher zu gestalten. Diese Methoden können beim `method`-Schlüssel abgerufen werden, der in Folienumgebungen und dem `\pdsetup`-verfügbar ist (siehe Abschnitt 2.2.2).

`method=normal`

Dies ist die vorgegebene Methode der Aufbereitung von Folien. Sie ist schnell und erlaubt die Verwendung von Overlays, nicht aber die wortwörtliche Wiedergabe.<sup>8</sup>

`method=direct`

Diese Methode ist auch schnell, aber sie erlaubt nicht die Verwendung von Overlays. Overlays werden unauffällig außer Gefecht gesetzt. Allerdings ermöglicht diese Methode wortwörtlichen Inhalt auf Folien.

`method=file`

Diese Methode nutzt einen provisorischen Ordner, um den Folienkörper zu exportieren und wieder einzulesen. Damit erlaubt sie die Verwendung von Overlays und wortwörtlichen Inhalt, aber sie ist mitunter langsamer, wenn viele Folien mit dieser Methode aufbereitet werden, da das Ordnersystem dann intensiv genutzt ist.

Hier ist ein Beispiel, dass die Nutzung aller drei Folienaufbereitungsmethoden demonstriert:

```
\documentclass{powerdot}
\usepackage{listings}
\lstnewenvironment{code}{
  \lstset{frame=single,escapeinside=' ',
    backgroundcolor=\color{yellow!20},
    basicstyle=\footnotesize\ttfamily}
}{}
\begin{document}
\begin{slide}{Slide 1}
Normal \pause content.
\end{slide}
\begin{slide}[method=direct]{Slide 2}
Steps 1 and 2:
\begin{code}
compute a;\pause'
compute b;
\end{code}
\end{slide}
\begin{slide}[method=file]{Slide 3}
Steps 1 and 2:
\begin{code}
compute a;\pause'
compute b;
\end{code}
\end{slide}
\end{document}
```

Die erste Folie zeigt das Standardverhalten für normalen Inhalt, sie produziert zwei Overlays. Trotz der Verwendung des `\pause`-Befehls erstellt die zweite Folie keine

<sup>7</sup>Und anderer Inhalt, der bei der Verarbeitung `catcode`-Veränderungen benötigt wird.

<sup>8</sup>Außer wenn es in einer Box außerhalb der Folie gesichert wurde.

Overlays. Dieser Befehl wurde untauglich gemacht mit der Wahl für die `direct` Methode, um wortwörtlichen Inhalt aufzubereiten. Die dritte Folie hat das gleiche Aussehen wie die zweite, aber jetzt werden zwei Overlays produziert, weil die Methode, die einen provisorischen Ordner nutzt, verwendet wurde. `\pause` wurde hier innerhalb der Auflistung gebraucht, der Befehl kann aber genauso außerhalb von Listenumgebungen benutzt werden.

## 6.5 Der `\twocolumn`-Befehl

`\twocolumn` Das `\twocolumn`-Makro erlaubt, den Inhalt auf zwei Spalten aufzuteilen.

```
\twocolumn[options]{left}{right}
```

Es wird links *left* und rechts *right*, also in zwei Spalten gesetzt. Die Abmessungen dieser Spalten können mit Optionen *options* kontrolliert werden. Hier die verfügbaren Optionen:

*option*  
lineheight

`lineheight`

Wenn `lineheight` spezifiziert wird, erscheint durch `\psline` eine Linie von bestimmter Höhe zwischen den Spalten. Beispiel: `lineheight=6cm`.

*option*  
lineprop

`lineprop`

Mit jeder `pstricks`-Angabe können die Linienproportionen näher bestimmt werden. Beispiel:

```
lineprop={linestyle=dotted,linewidth=3pt}
```

*options*  
lfrheight  
lfrprop

`lfrheight` `lfrprop`

Ersteres schafft einen Rahmen von bestimmter Höhe um die linke Spalte, Zweites ist wie `lineprop`, aber für den linken Rahmen.

*options*  
rfrheight  
rfrprop

`rfrheight` `rfrprop`

Genau wie `lfrheight` und `lfrprop`, allerdings für den rechten Rahmen.

*options*  
lcolwidth  
rcolwidth

`lcolwidth` `rcolwidth`

Die Breite von linker und rechter Spalte. Beide sind vorgegeben mit  $0.47\backslashlinewidth$ .

*option*  
frsep

`frsep`

Platz zwischen Text und Rahmen, vorgegeben mit  $1.5\text{mm}$ .

*option*  
colsep

`colsep`

Platz zwischen den beiden Spalten, vorgegeben mit  $0.06\backslashlinewidth$ .

*option*  
topsep

`topsep`

Der extra Platz (zusätzlich zu `\baselineskip`) zwischen dem Text über den Spalten und dem Text in den Spalten, vorgegebener Wert:  $0\text{cm}$ .

*option*  
bottomsep

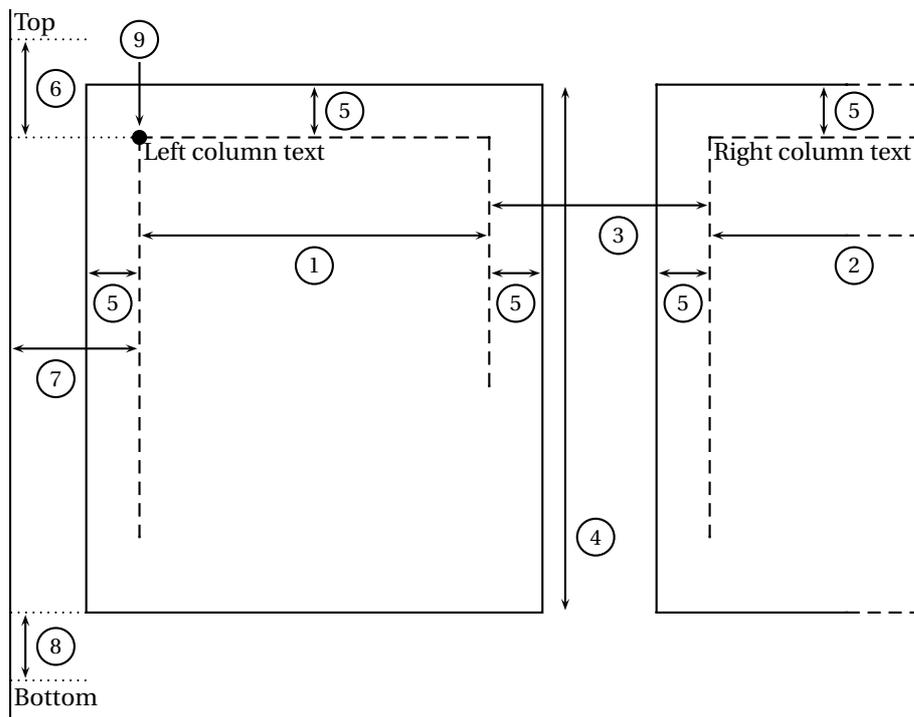
`bottomsep`

Der extra Platz zwischen dem Text in den Spalten und dem Text darunter, vorgegebener Wert:  $0\text{cm}$ .

*option*  
indent

`indent`

Horizontaler Zeileneinzug links zur linken Spalte, vorgegebener Wert:  $0\text{cm}$ .



Bedeutung der Labels

1	lcolwidth	5	frsep
2	rcolwidth	6	topsep
3	colsep	7	indent
4	lfrheight, rfrheight, lineheight	8	bottomsep
		9	Referenzpunkt

Abbildung 1: Zwei-Kolumnen-Dimensionen

Die oben beschriebenen Abmessungen sind in Abbildung 1 graphisch dargestellt.

Es ist wichtig zu bemerken, dass das `\twocolumn`-Makro die aktuelle Cursor-Position als Bezugspunkt für die erste Zeile des Textes der linken Spalte nutzt (siehe auch Figur 1). Das heißt, dass der optionale Rahmen bis zum Text der vorausgehenden Zeile ausgedehnt werden kann. Man kann in diesem Fall beispielsweise `topsep=0.3cm` angeben, um extra Platz zwischen diesen beiden Textzeilen zu schaffen. Der vorgegebene `topsep`-Wert basiert darauf, dass sich kein Text unmittelbar über den Spalten befindet. In diesem Fall lokalisiert man am besten den Ort der ersten Textzeile der linken Spalte am gleichen Punkt, bei dem ein gewisser Text nicht durch den `\twocolumn`-Befehl auf anderen Folien erstellt wird. Die `topsep=0cm`-Einstellung bewirkt genau das. Durch die Kombination von `topsep` und `indent` lässt sich dieses Verhalten und die Position der ersten Textzeile der linken Spalte beliebig ändern.

Das `\twocolumn`-Makro errechnet die Größe der Konstruktion, um den Text darunter korrekt zu positionieren. Die Berechnung ist fertig, wenn für `lfrheight`, `rfrheight`, `lineheight` (falls genauer bestimmt) das Maximum festgesetzt wird. So werden die rechte und linke Spalte stimmig ausgegeben. Wenn weder Rahmen noch Linien eingefügt werden, setzt `bottomsep` den horizontalen Platz zwischen der untersten Textzeile der Spalten und dem Text unter den Spalten (zusätzlich zu `\baselineskip`). Hier ein Beispiel:

```
\begin{slide}{Two columns}
Here are two columns.
```

```

\twocolumn[
  lfrprop={linestyle=dotted,linewidth=3pt},
  lfrheight=4cm,rfrheight=5cm,lineheight=3cm,topsep=0.3cm
]{left}{right}
Those were two columns.
\end{slide}

```

Hier könnte die Verwendung der xkeyval-Befehle `\savevalue` und `\usevalue` nützlich sein, zum Beispiel wenn man die Eigenschaften des linken Rahmens für den rechten kopieren will. So vermeidet man nicht nur eine überflüssige zweite Typisierung, sondern auch Fehler zu machen, die in unterschiedlich großen Rahmen resultieren würden. Betrachten wir das unten stehende Beispiel:

```

\twocolumn[
  \savevalue{lfrheight}=3cm,
  \savevalue{lfrprop}={
    linestyle=dotted,framearc=.2,linewidth=3pt},
  rfrheight=\usevalue{lfrheight},
  rfrprop=\usevalue{lfrprop}
]{left}{right}

```

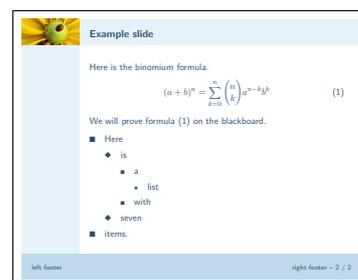
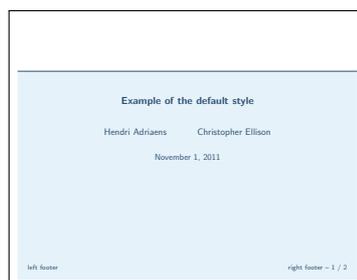
Ziehen Sie die xkeyval-Dokumentation [2] zu Rate, um mehr über die `\savevalue`- und `\usevalue`-Befehle zu erfahren.

## 7 Zur Verfügung stehende Stile

powerdot enthält eine Zahl von Stilen, welche im nachfolgenden Überblick aufgeführt sind. Die Charakteristik jedes Stils ist kurz beschrieben und von einem Beispiel einer Titelfolie und einer normalen Folie begleitet. Stile, die auf der widenslide-Umgebung beruhen, haben ein Inhaltsverzeichnis auf dem linken Rand im Querformat (landscape) und am unteren Rand im Hochformat (portrait) der Folien. Das Hochformat wird unterstützt, sofern es nicht anders angelegt wird.

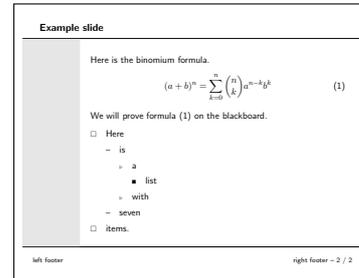
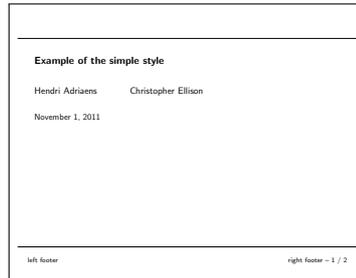
### default

Dieser Standard-Stil bietet sechs verschiedene Layouts. Jedes wird von einer Blume in der linken, oberen Ecke dekoriert. Das blaue Layout des default-Stils hat als Hauptfarben hellblau, blau und weiß, wie im Beispiel unterhalb. Andere verfügbare Layouts sind red, green, yellow, brown and purple.



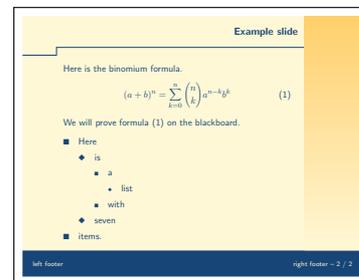
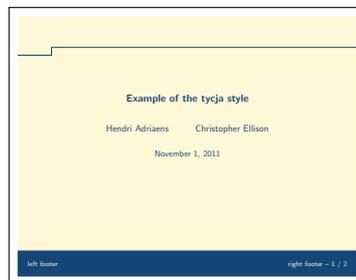
simple

Dies ist ein einfacher Stil in schwarz und weiß. Er könnte für Folien nützlich sein, die ausgedruckt werden sollen.



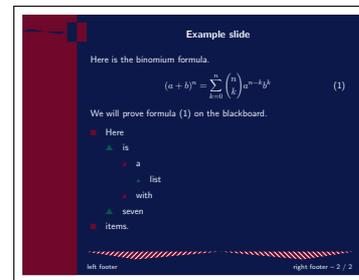
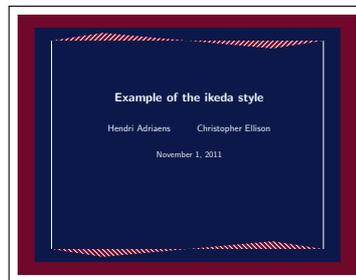
tycja

Dieser Stil zeigt sich in gelben und dunkelblauen Schattierungen. Im Querformat ist das Inhaltsverzeichnis auf der rechten Seite und im Hochformat am unteren Rand der Folien.



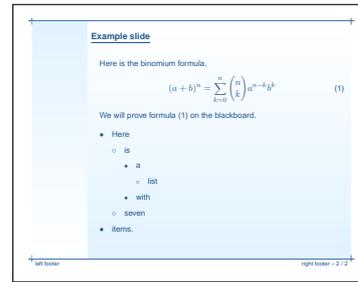
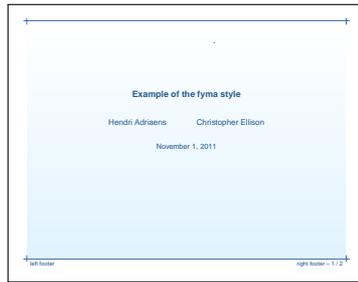
ikedda

In diesem Stil zeigen sich die Folien in dunklen Schattierungen von Rot und Blau und dazu in heller Textfarbe. Zudem wird ein schmuckvolles Muster auf der Folie verwendet.



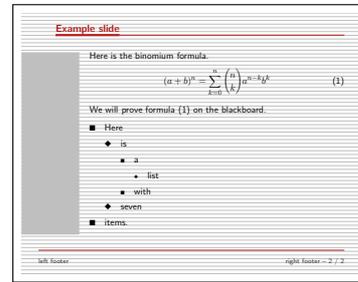
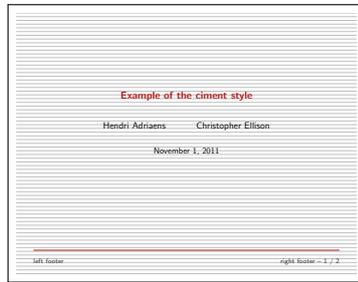
fyma

Dieser Stil wurde von Laurent Jaques für prosper entworfen und darauf basierend, entwickelte er eine Version für HA-prosper mit zusätzlichen Eigenschaften. Mit freundlicher Genehmigung wurde der Stil von Shun'ichi J. Amano konvertiert, um ihn auch für powerdot verfügbar zu machen. Das Layout blue hat ein elegantes Design mit einem Verlauf aus hellblau und weiß als Hintergrund (siehe Beispiel). Weitere Layouts sind green, gray, brown und orange. Außerdem gibt es spezielle Schablonen für Bereiche auf Folien und breite Folien.



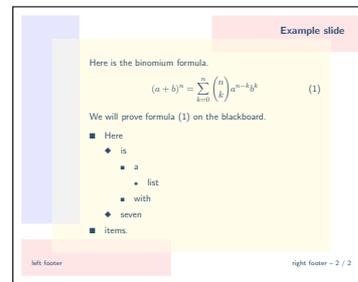
ciment

Entworfen von Mathieu Goutelle für prosper und HA-prosper ist auch dieser Stil dank freundlicher Genehmigung umgewandelt worden und somit für powerdot verfügbar. Der Stil hat einen mit hellgrauen Linien, horizontal schraffierten Hintergrund. Inhaltsverzeichnis und Überschriften sind mit einem dunklen Rot hervorgehoben.



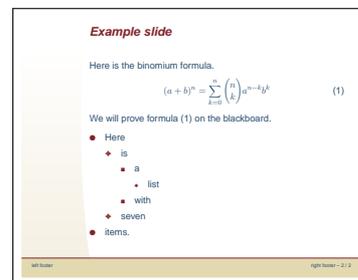
elcolors

Dieser Stil benutzt helle Farbnuancen der Grundfarben rot, blau und gelb.



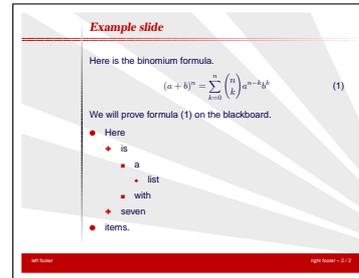
aggie

Jack Stalnaker entwarf diesen Stil für HA-prosper und hat ihn dann für powerdot konvertiert. Verwendet wird dunkles Rot und ein helles Braun.



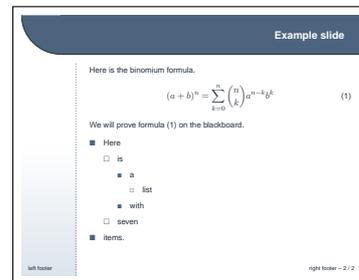
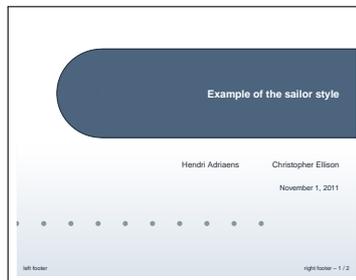
husky

Auch dieser Stil stammt von Jack Stalnaker und zeigt eine markante rote Schrift vor einem sonnenartigen, hellgrauen Hintergrund.



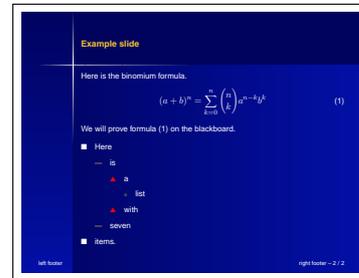
sailor

Dieser Stil wurde von Mael Hilléreau beigetragen und bietet fünf verschiedene Layouts: Sea (the default), River, Wine, Chocolate und Cocktail. Abgebildet ist ein Beispiel des Layouts Sea.



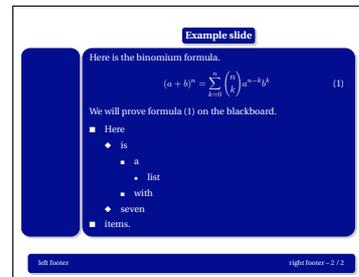
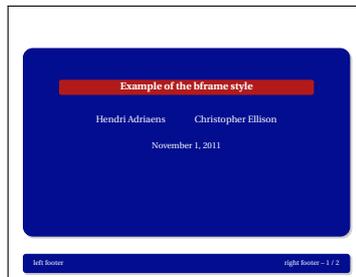
open

Ein tiefes Blau als Hintergrund und gelber Text zeichnen diesen Stil aus. Er stammt von Piskala Upendran.



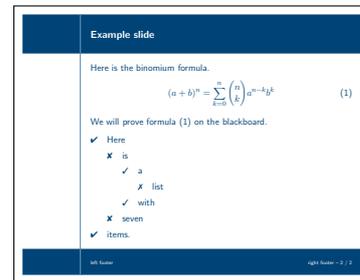
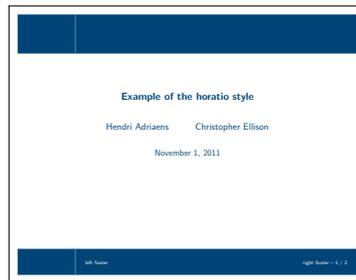
bframe

Der bframe Stil ist ebenfalls von Piskala Upendran und hat blaue, abgerundete Boxen in welche der weiße Text eingefügt wird.



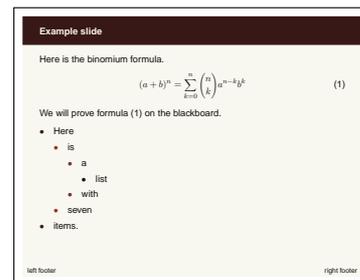
## horatio

Der Stil horatio wurde von Michael Lundholm beigesteuert und ist eher zurückhaltend, in blauer und weißer Farbgebung.



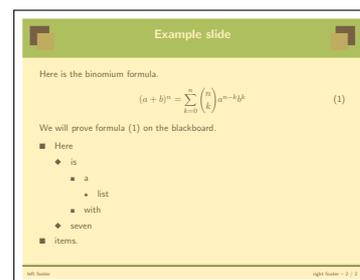
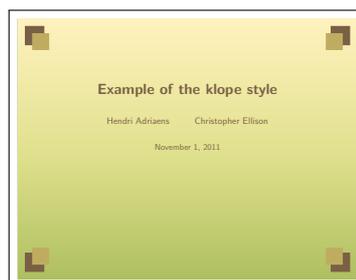
## paintings

Dies ist ein einfacherer Stil ohne Inhaltsverzeichnis. Er wurde von Thomas Kopsell entworfen und ermöglicht 10 verschiedene Layouts. Die dabei jeweils verwendeten Farben lehnen sich an berühmte Gemälde an.<sup>9</sup> Welche Gemälde als Vorlage dienten, kann man in der Stildatei, der den Stil definierenden Datei nachlesen. Die zum Stil verfügbaren Layouts lauten: Syndics (Standard-Layout), Skater, GoldenGate, Lamentation, HolyWood, Europa, Moitessier, MayThird, PearlEarring und Charon. Hier ist ein Beispiel in Syndics.



## klope

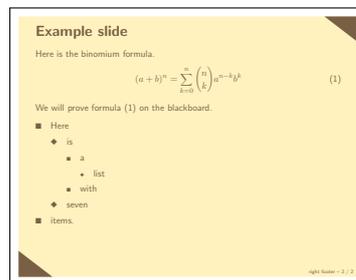
Der Stil klope führt ein horizontales Inhaltsverzeichnis aus, das nur die als section deklarierten Gliederungspunkte auflistet. Folgende Layouts stehen zur Verfügung: Spring, PastelFlower, BlueWater und BlackWhite. Das Layout Spring, welches hier als Beispiel dient, entspricht dem Standard-Layout.



<sup>9</sup>Der Stil benutzt einen Farbton, `pdcolor7`, welcher nicht in den Layouts verwendet wird, aber dennoch wie die anderen Farben von entsprechenden Gemälden stammt. Der Farbton kann beispielsweise benutzt werden, um einen Text hervor zu heben.

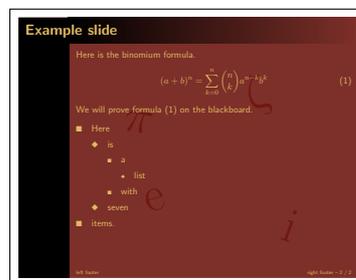
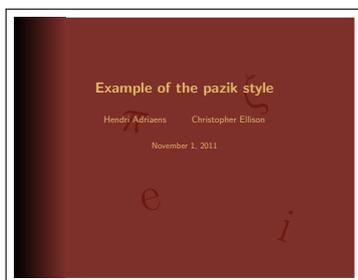
jefka

Dieser Stil bietet vier Layouts: brown (Standard), seagreen, blue und white. Die Beispielfolien entsprechen dem Layout brown.



pazik

Dieser Stil ist in zwei Layouts verwendbar: brown und auch red, wie in der Beispielabbildung.



## 8 Kompilieren der Präsentation

### 8.1 Abhängigkeiten

In der Tabelle 2 ist eine Liste von Dateipaketen aufgeführt, die powerdot verwendet, um spezifische Aufgaben auszuführen. Abhängigkeiten der jeweiligen Pakete sind in dieser Tabelle jedoch nicht mit aufgeführt. In der Tabelle meint ‘required’ (erforderlich), dass das verwendete Paket *mindestens* so aktuell sein sollte, wie die gelistete Version und ‘tested’ (getestet) meint, dass powerdot mit dieser Paketversion getestet wurde, aber es auch mit älteren oder neueren als der gelisteten Version funktionieren könnte. Tritt bei der Programmumwandlung ein Fehler auf, empfiehlt es sich, zur Behebung als erstes die Anforderungen an die benötigten Paketversionen zu prüfen. Um zu erfahren, welche Version eines Pakets aktuell verwendet wird, hilft es den Befehl `\listfiles` in die erste Zeile des  $\LaTeX$  Quelltextes zu setzen, die `.log` Datei zu öffnen und die Dateiliste zu lesen (siehe  $\LaTeX$ -Handbuch für mehr Informationen). Nötige Paket-Updates zur Aktualisierung sind bei CTAN [7] erhältlich.

### 8.2 Erzeugen und Ansehen der Output-Datei

Um die Präsentation zu kompilieren, muss die Datei mit  $\LaTeX$  laufen. Die DVI, die produziert wird, kann mit dem DVI-Viewer YAP<sup>10</sup> von MiKTeX angezeigt werden. Leider werden von xdvi und kdvi (kile) nicht alle PostScript-Besonderheiten unterstützt. Daher werden diese die Präsentation inkorrekt anzeigen. Falls der verwendete DVI-Viewer dies jedoch dennoch unterstützt, sollte sicher gestellt werden, dass die DVI-Anzeigeeinstellungen den Einstellungen der Präsentation angepasst sind. Im Falle der Verwendung von screen-Format sollte in den DVI-Anzeigeeinstellungen das `letter`

<sup>10</sup>Mit Ausnahme der Verwendung von `pstricks-add`, welches das DVI-Koordinatensystem verfälscht.

Package/file	Version	Date	Required/tested
xkeyval [2]	2.5c	2005/07/10	required
pstricks.sty [16, 17]	0.2l	2004/05/12	required
xcolor [10]	1.11	2004/05/09	required
enumitem [4]	1.0	2004/07/19	required
article class	1.4f	2004/02/16	tested
geometry [15]	5.x	2010/10/07	tested
hyperref [14]	6.74m	2003/11/30	tested
graphicx [5]	1.0f	1999/02/16	tested
verbatim	1.5q	2003/08/22	tested

Tabelle 2: Abhängigkeiten

bzw. Dokumentformat gewählt werden. Falls der DVI-Viewer Benutzereinstellungen für das Seitenformat zulässt, ist die Einstellung von 21 mal 28 cm zu verwenden.

Es sei darauf hingewiesen, dass bestimmte Projekte, die mit PostScript- oder PDF-Programmen erstellt wurden, mit dem DVI-Viewer nicht kompatibel sind. Beispielsweise werden im Gegensatz zur PostScript-Ausgabe nicht alle Elemente angezeigt (wie bei `\pause`, siehe Kapitel 4) oder es gibt fehlende Verlinkungen im Inhaltsverzeichnis.

Wenn man ein PostScript-Dokument erzeugen möchte, startet man die DVI mit `dvips` *ohne eine Einstellung bezüglich des Formats oder der Seitengröße*. `powerdot` wird dann die nötige Information in der DVI-Datei ergänzen, `dvips` und `ps2pdf` (`ghostscript`) hilft, ein angemessenes Dokument zu erzeugen. Sollte dies aus bestimmten Gründen nicht funktionieren oder man möchte das Seitenformat selbst spezifizieren, sollte man die Option `nopsheader` verwenden, welche im Abschnitt 2 beschrieben wird. Das PostScript-Dokument könnte schließlich dazu dienen, mit Unterstützung des `psnup`-Hilfsprogramms mehrere Folien auf eine Seite zu bringen.

Um eine PDF-Datei für eine Präsentation zu erstellen, kompiliert man die mit `dvips` erstellte PS-Datei mit `ps2pdf` zum PDF-Dokument. Auch hier *muss keine Angabe für Format oder Seitengröße angegeben werden*. Treten dabei Probleme auf, ist wieder die Option `nopsheader` hilfreich, um selber Formateinstellungen vornehmen zu können.

## 9 Einen eigenen Foliensstil erstellen

### 9.1 Generelle Informationen

Es ist nicht schwierig, `powerdot`-Stilvorlagen zu bearbeiten oder selbst zu erstellen. Wenn man einen Stil modifizieren oder einen neuen entwerfen will, muss man zunächst die Datei im  $\text{T}_{\text{E}}\text{X}$ -Baum finden, die als Basis verwendet werden soll. Diese Stildatei ist mit `powerdot-<style_name>.sty` zu benennen<sup>11</sup>, auch um Namenskonflikte zu umgehen. Der neu benannte Stil muss dann in dem lokalen  $\text{T}_{\text{E}}\text{X}$ -System ergänzt werden, um ihn verwenden zu können (für mehr Informationen siehe die  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ -Distribution).

Nachdem dies beachtet wurde, kann der neue Stil entworfen werden. Dazu ist es äußerst hilfreich, die Stildateien zu studieren (bspw. `powerdot-default.sty`) und den Rest dieses Kapitels zu lesen, er bietet gute Beispiele zum Text.

Ein Stil hat verschiedene Komponenten, welche nun beschrieben werden sollen.

<sup>11</sup>Die  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  Public Project License fordert die Umbenennung von modifizierten Dateien. Siehe auch <http://www.latex-project.org/lppl>.

## Identifikation und Pakete

So identifiziert das Paket im Protokoll der Präsentation und lädt alle benötigten Pakete. Der Standardstil default beinhaltet so etwas wie:

```
\NeedsTeXFormat{LaTeX2e}[1995/12/01]
\ProvidesPackage{powerdot-default}[2005/10/09 v1.2 default style (HA)]
\RequirePackage{pifont}
```

Mehr Informationen über diese Befehle findet man im  $\LaTeX$ -Handbuch [12].

## Layouts oder Farbwahl

Dieser Abschnitt enthält die Definition der Layouts der Paletten oder Farben, die in einem Stil verwendet werden. powerdot verwendet xcolor (über pstricks). Daher eignet sich für mehr Informationen zu Farben die xcolor-Dokumentation. Genauere Ausführungen über Layouts finden Sie zudem im Kapitel 9.2.

## Definition von Templates

Hiermit beschäftigen sich die Kapitel 9.3 bis 9.9.

## Benutzereinstellungen

Alles was Teil des Stils sein soll, kann hier eingeschlossen sein. Der default-Stil beinhaltet beispielsweise die Definition für die Symbole in Listen, wie `itemize`, sowie einige Bereitstellungen für Listen generell (umgesetzt mit `\pdsetup`, siehe Kapitel 2.2). In diesem Teil der \*.sty-Datei können also Benutzereinstellungen vorgenommen werden, wie in Kapitel 9.10 weiter beschrieben wird.

## Bereitstellung der Schriftart

Dies setzt das Schriftdesign fest, welches durch Laden von Paketen umgesetzt werden kann, wie beispielsweise helvet.

## 9.2 Paletten definieren

Die ausdrückliche Definition von Mustervorlagen wird in Kapitel 9.3 thematisiert. Hier sei zunächst nur verdeutlicht, dass die Vorlage die Schablone für den Stil einer Folie ist und somit ihr formales Design ausmacht. Paletten hingegen sind Farbgruppen, in denen ein Stil ausgeführt werden kann. Eine Palette ändert demnach nicht das gesamte Design eines Musters. Der folgende Befehl dient der Definition von Paletten für einen eigenen Stil.

`\pddefinepalettes`

```
\pddefinepalettes{<nameI>}{<cmdsI>}...
```

Dieses Makro (ein Programmcodeteil) funktioniert mit *jeder* geraden Zahl obligatorischer Argumente von mindestens zwei. Für jeden Namen *<name>* eines Layouts ist eine Gruppe von Befehlen *<commands>* zur Definition möglich. Diese Befehle können die Farben mit Namen wie `pdcolor1`, `pdcolor2`, etcetera definieren. Diese Farben können beim Entwerfen eines Musters verwendet werden (sehen Sie Kapitel 9.3). `pdcolor1` entspricht dabei der Textfarbe.

Der Benutzer kann auf diese Paletten mit dem `palette`-Schlüssel im `\pdsetup`-Befehl zugreifen (sehen Sie Kapitel 2.2). Wenn der Nutzer keine Palette spezifiziert, wird die erste definierte Palette beim Kompilieren genutzt. Hier nun ein Beispiel zur Definition von zwei Paletten.

```
\pddefinepalettes{reds}{
\definecolor{pdcolor1}{rgb}{1,0,0}
\definecolor{pdcolor2}{rgb}{1,.1,0}
\definecolor{pdcolor3}{rgb}{1,.2,0}
}{greens}{
\definecolor{pdcolor1}{rgb}{0,1,0}
```

```

\definecolor{pdcolor2}{rgb}{.1,1,0}
\definecolor{pdcolor3}{rgb}{.2,1,0}
}

```

In diesem Beispiel dient die red-Palette die Standardfarbgebung für den Stil. Mehr Informationen über `\definecolor` konsultieren Sie die Dokumentation zum `xcolor`-Paket [10].

Die Verwendung der Farbnamen `pdcolor2`, `pdcolor3` etcetera ist nicht zwingend nötig. Diese Farben sind jedoch definiert. `powerdot` verwendet `random dots` (zufällige Punkte) (sehen Sie Kapitel 2.2.2). Die Flexibilität ermöglicht weitere Festlegungen für Vorlagen von Folien und deren Farbpaletten. Ein Beispiel, wie die Möglichkeiten weiter auszuschöpfen sind, bietet ein Einblick in den Stil `klope`.

### 9.3 Definition von Vorlagen

Wir beginnen mit einer Definition von Vorlage (Template). Die Vorlage umfasst eine Gruppe von Festlegungen für Elemente der Folie als auch Festlegungen des Benutzers, welche den visuellen Charakter der Präsentation bestimmen. Ein Stil kann mehrere Formelemente enthalten.

```

\pdefinemplate [basis] {name} {options} {commands}

```

`\pdefinemplate`

Dies definiert die Umgebung *name*, welche eine Vorlage, bestimmt durch die Charakteristika *basis*, *options* und *commands*. Diese Elemente werden in den nächsten Kapiteln genauer erläutert.

Sollen verschiedene Vorlagen erstellt werden, die sich nur gering voneinander unterscheiden, lohnt es, eine *basis*-Schablone zu definieren, von welcher aus die anderen Varianten erstellt werden. Alle *options* und *commands* für die neue Vorlage *name* werden der bestehenden Liste von *options* und *commands* der *basis*-Schablone angefügt.

Die Vorlage sollte mit *einem passenden Namen* benannt, die Neubenennung bestehender Schablonen oder Umgebungen jedoch vermieden werden. `powerdot` definiert `pauseslide`, `note` und `emptyslide` intern, wobei eine Verwendung dieser Namen in der Regel vermieden werden sollte. Außerdem besteht jeder Designstil aus den Vorlagen `slide` und `titleslide`. Die `titleslide`-Umgebung wird im Standardfall dazu verwendet, die Titelfolie zu erzeugen, während `slide` (standardmäßig) für Unterkapitel verwendet wird. Titel und Unterkapitel benutzen die *options* auf besondere Weise, was detaillierter in Kapitel 9.10 erläutert ist.

### 9.4 Überwachung des Setup

*option*  
`ifsetup` Nachfolgend sind die *options* (Schlüssel bzw. Parameter) beschrieben. Sie können mit dem Key `ifsetup` überprüfen, wie die verschiedenen Setups angewendet werden. Jeder verwendete Key, der vor der ersten `ifsetup`-Deklaration in *options* auftritt, wird bei jedem möglichen Setup befolgt. Die dem Key zugewiesenen Parameter werden umgesetzt. Wird der `ifsetup`-Key jedoch einmal verwendet, erfolgt die Anwendung der untergeordneten Keys *nur* auf das deklarierte Setup im `ifsetup`-Key. Die Parameter untergeordneter Keys werden demnach nur für das zugewiesene Setup beachtet. Dabei kann der `ifsetup`-Key mehrere Male verwendet werden.

Mit möglichen Setups sind die zulässigen Werte für die Optionen gemeint, wie passende Werte für die Parameter `mode`, `paper`, `orient` und `display`, beschrieben in dem Kapitel 2.1. Wenn ein Wert (oder Werte!) für einen dieser vier Keys nicht in einer `ifsetup`-Zuweisung spezifiziert ist, werden alle untergeordneten Key-Deklarationen zu einem beliebigen Layout diesen Typs angewendet. Man betrachte folgenden Quelltext als Beispiel.

```

1 ...
2 textpos={.2\slidewidth,.3\slideheight},
3 ifsetup={portrait,screen},
4 textpos={.3\slidewidth,.2\slideheight}
5 ...
6 ifsetup=landscape,
7 ...
8 ifsetup,
9 ...

```

Angenommen, im Beispiel wäre keine `ifsetup`-Deklaration vor der ersten `textpos`-Deklaration, würde der Befehl `textpos` auf jedes mögliche Setup angewendet werden. Für das `screen`-Format in Hochformat (`portrait`) jedoch, wird die nächste `textpos`-Deklaration befolgt. Das heißt, alle Befehle, die bis zum nächsten `ifsetup` (das alle Papiersorten spezifiziert und die `landscape`-Orientierung) ausgeführt werden, werden für das `portrait-screen`-Layout umgesetzt. Alle Keys nach der nächsten `ifsetup`-Deklaration gelten für Querformate (`landscape orientation`), wobei *paper*, *mode* und *display* nicht spezifiziert sind. Wenn man nach einer Spezifikation von Keys zu den Optionen zurück möchte, die auf alle Setups angewendet werden, deklariert man `ifsetup` als empty, ohne weitere Parameter, so wie im Beispiel. Alle nachfolgenden Deklarationen werden nun wieder für jedes mögliche Setup befolgt.

Der folgende Befehl ist eine unabhängige Anwendung des zuvor beschriebenen Mechanismus. Er erlaubt die Kontrolle des Setups außerhalb des Arguments `<options>` des `\pddefinemplate`-Befehls.

```
\pdifsetup{<desired>}{<true>}{<false>}
```

`\pdifsetup` Dieser Makro entscheidet `<true>`, wenn die Programmeinrichtung des Benutzers, sein Setup, mit dem verlangten `<desired>`-Setup übereinstimmt. In allen anderen Fällen gilt `<false>`. Wurde beispielsweise `landscape` (Querformat) gewählt, dann gilt

```
\pdifsetup{landscape}{yes}{no}
```

`yes`. Wurde jedoch statt dessen `portrait` (Hochformat) gewählt, würde `no` gelten.

Dieser Makro kann genutzt werden um die Setupanforderungen zu überprüfen und um zum Beispiel einen Fehler zu generieren, wenn eine bestimmte Programmeinrichtung vom Stil nicht unterstützt wird. `powerdot` bietet eine vorverfasste Fehler-Nachricht, welche in den ersten Zeilen des Quelltextes ihrer Stildatei verwendet werden kann.

```
\pd@noportrait
```

`\pd@noportrait` Dieses Makro generiert einen Fehler, wenn der Benutzer Hochformat abfragt. An dieser Stelle sei darauf hingewiesen, dass ein eventueller Handout-Modus nur in Hochformat möglich ist. Dieses Makro berücksichtigt dies jedoch und generiert bei Abfrage eines Handouts keinen Fehler.

## 9.5 Hauptkomponenten

Die `<options>` kontrollieren verschiedene Key-Komponenten einer Folie. Jede Komponente hat verschiedene Einzelteile mit jeweiligen Eigenschaften. Ein Key, der für das `<options>`-Argument verwendet werden kann, ist die Bezeichnung der Komponente. Zum Beispiel benannt nach ihrer Eigenschaft, die kontrolliert werden soll.

Die Komponenten `title`, `text`, `toc`, `stoc` und `ntoc` haben die Einzelteile bzw. Eigenschaften `hook`, `pos`, `width` und `font`. Zudem hat die Komponente `text` die Eigenschaft `height`. Die Komponenten `lf`, `cf` und `rf` haben `hook`, `pos`, `temp` und `font`. Daher sind `titlefont`, `tocpos` und `lftemp` Beispiele mit verschiedenen Keys. Diese Komponenten und ihre Einzelteile sollen nun erläutert werden.

Nun ein Überblick über die Komponenten, die vom  $\langle options \rangle$ -Argument im Template `\pddefinemplate` kontrolliert werden.

- option* `title-`  
*title-* Betrifft den Titel der Folie.
- option* `text-`  
*text-* Betrifft die Box für den Haupttext auf der Folie.
- option* `toc-`  
*toc-* Betrifft das vollständige Inhaltsverzeichnis einschließlich der Unterkapitel und Folien.
- option* `stoc-`  
*stoc-* Dieses Inhaltsverzeichnis umfasst nur die Unterkapitel „sections“ (sehen Sie auch `ntoc`).
- option* `ntoc-`  
*ntoc-* Dies ist ein Inhaltsverzeichnis, welches nur die Einträge des aktuellen Kapitels anzeigt. Zusammen mit `stoc` kann es als ein gesplittetes Inhaltsverzeichnis verwendet werden. Beispielsweise mit Überblick über die Kapitel im einen und mit den Unterpunkten des jeweiligen Kapitels im anderen Verzeichnis. In einer einzelnen Folienvorlage ist `toc`, man würde üblicherweise ein Inhaltsverzeichnis haben, eine Kombination von `stoc` und `ntoc` oder überhaupt kein Inhaltsverzeichnis.

- options* `lf-` `cf-` `rf-`  
*lf-* *cf-* *rf-* Betrifft die linke, mittlere und rechte Fußzeile.
- Beachten Sie, dass alle Positionierungen von Komponenten intern über `\rput` von `pstricks` eingebracht werden [16, 17]. In der `pstricks`-Dokumentation finden Sie weitere Informationen zu diesem Befehl. Zudem sollte zur Kenntnis genommen werden, dass alle Komponenten (außer `lf`, `cf` und `rf`) ihren Inhalt in eine `minipage`-Umgebung setzen.

Nun werden alle Einzelteile bzw. Eigenschaften der zuvor genannten Komponenten mit ihrer Bedeutung aufgeführt. Es ist daran zu denken, dass Keys aus der Kombination einer Komponente und einer Eigenschaft bestehen.

- option* `-hook`  
*-hook* Diese Option definiert den `\rput`-Anker, welcher bei der Positionierung eines Item verwendet wird. Dieser kann sein: `t1`, `t`, `tr`, `r`, `Br`, `br`, `b`, `b1`, `B1`, `l`, `B` und `c`. Die `pstricks`-Dokumentation bietet hierzu weitere Informationen.

- option* `-pos`  
*-pos* Dies definiert die Position des `hook` der Seite. Die linke, untere Ecke des „Papiers“ entspricht dem Punkt  $\{0,0\}$  und die rechte, obere Ecke dem Punkt  $\{\slidewidth, \slideheight\}$ . Soll die Box für den Haupttext 20% vom linken Rand und 30% vom oberen Rand entfernt sein, ergibt sich folgender Key:

```
textpos={.2\slidewidth,.7\slideheight}
```

Wenn die Position einer Komponente nicht spezifiziert wurde, wird die Komponente nicht auf der Folie platziert. Dies ermöglicht es, Folien zu entwerfen, die in einem nüchtern Stil ohne Fußnoten oder Inhaltsverzeichnis gehalten sind.

*option*  
-width

**-width**

Bezieht sich auf die Breite der Komponente. Alle Komponenten, die powerdot positioniert, werden in eine minipage-Umgebung gesetzt. Die Eigenschaft width bestimmt die Breite der minipage. Beispielsweise:

```
textwidth=.7\slidewidth
```

Für lf, cf and rf-Komponenten gibt es diese Eigenschaft nicht.

*option*  
-height

**-height**

Diese Option ist nur für die Komponente text verfügbar. Oder anders: Für diese Eigenschaft gibt es nur eine Verwendung, nämlich den Key textheight. Damit kann die Höhe der minipage spezifiziert werden, welche als Haupttextfenster dient. Diese Höhe dient jedoch nur für die vertikale Angleichung von Material auf der Folie, wie beispielsweise Fußzeilen; nicht aber für die Länge oder den automatischen Folienumsprung durch powerdot bei vollen Textfenstern. Der vorbestimmte Wert ist \slideheight.

*option*  
-font

**-font**

Dies wird im Quelltext kurz vor den betroffenen Text gesetzt, dessen Schriftsatz definiert werden soll. Mit dem Befehl können so Abweichungen des Textes in Schriftart und Farbe vorgenommen werden. Dies kann als Schriftsatzdeklaration umgesetzt werden, wie \large\bfseries, aber auch mit anderen Inhalten wie \color{red} oder \raggedright.

*option*  
-temp

**-temp**

Diese Eigenschaft ist nur für die Fußzeilen (lf, cf und rf) verfügbar und kann verwendet werden, um deren Schablone in der Vorlage zu ändern. Beispielsweise kann Inhalt durch den Benutzer hinzugefügt werden. Die Standarddeklaration bei powerdot ist folgende:

```
rftemp=\pd@@rf\ifx\pd@@rf\@empty  
\else\ifx\theslide\@empty\else\ -- \fi\fi\theslide
```

\pd@@rf wird hier den Inhalt des rechten Fußfeldes umfassen, definiert durch den Benutzer mit dem \pdsetup-Befehl. Ähnlich beinhaltet \pd@@lf den Inhalt des linken Fußfeldes. Der obige Befehl kontrolliert, ob das rechte Fußfeld und \theslide Inhalte haben, nicht leer sind. Wenn dem so ist, wird -- eingesetzt, um sie zu unterscheiden.

*option*  
-orient

**-orient**

Diese Eigenschaft ist nur für die Komponenten toc, stoc und ntoc verfügbar. Es stehen die Alternativen h oder v zur Auswahl, um die horizontale oder vertikale Ausrichtung des Inhaltsverzeichnisses zu bestimmen. Die Voreinstellung entspricht v. Bezüglich dem Anlegen des Inhaltsverzeichnisses liefert Kapitel 9.6 weitere Informationen.

## 9.6 Das Folieninhaltsverzeichnis

Das kleine Inhaltsverzeichnis auf den Folien kann durch vier Makros und mehrere Optionen gesteuert werden.

\pd@tocslide  
\pd@tocsection

Diese Makros nehmen ein Argument an. Beim Erstellen des Inhaltsverzeichnisses durchläuft powerdot den Inhalt zunächst \pd@tocslide oder \pd@tocsection, je nach dem welcher Eingabetyp gerade erstellt wird. Sie können beispielsweise

```
\def\pd@tocslide#1{\bullet$ \ #1}
\def\pd@tocsection#1{#1}
```

eingeben, wodurch alle normalen Eingaben (nicht die Abschnitte) mit einem „Bullet“ präfigiert werden. Diese beiden Makros sind standardmäßig so definiert, dass sie sich genau auf ihre jeweiligen Argumente übertragen.

`\pd@tocdisplay`  
`\pd@tochighlight` Diese zwei Makros nehmen ebenfalls ein Argument. Nachdem die Eingabe mit dem Befehl `\pd@tocslide` oder `\pd@tocsection` bearbeitet wurde, setzt `powerdot` das Erstellen der Eingabe mit dem Durchlaufen von `\pd@tocdisplay` fort, wenn die Eingabe nur angezeigt werden muss oder von `\pd@tochighlight`, wenn die Eingabe hervorgehoben werden muss. Diese Makros sind überwiegend beteiligt und betreuen das Erstellen des Inhalts in angemessener Schriftart und Farbe in einer `minipage`. Des Weiteren setzt `\pd@tochighlight` eine Box um die Einheit.

Beachten Sie, dass beispielsweise beide, zum einem die Eingabe des gesonderten Inhaltsverzeichnis, genauso wie das Inhaltsverzeichnis als Ganzes in einer `minipage`-Umgebung von diesen Makros gesetzt sind, in dem Fall, dass das Inhaltsverzeichnis vertikal angelegt ist. Die `-width`-Bestandteile determinieren dann die Breite des Inhaltsverzeichnisses und zusammen mit `tocsecindent` und `tocslideindent` (siehe unten) die Breite der individuellen Eingaben. Wenn es horizontal ist, sind nur die gesonderten Eingaben und nicht das Inhaltsverzeichnis an sich in der `minipage`. Die `-width`-Bestandteile determinieren nur die Breite der individuellen Eingaben (zusammen mit `tocsecindent` und `tocslideindent`).

Mehrere Aspekte des Prozesses des Anlegens des Inhaltsverzeichnisses können durch Schlüssel gesteuert werden, die im Befehl `\pddefinemplate` abrufbar sind, die dann unten beschrieben werden. Falls diese Schlüssel nicht genug Spielraum bereitstellen, um tun zu können, was Sie möchten, müssen Sie vielleicht einen Blick auf die zwei Makros in der Quelle werfen und sich entscheiden diese in ihrem Stil neu zu schreiben, bis sie zu ihren Bedürfnissen passen. Ein Beispiel finden Sie im `fyma`-Stil.

*option*  
`tocfrsep`

`tocfrsep`

Diese Länge ist der Abstand zwischen der Box, die den Inhalt umgibt, der in der `minipage` erzeugt wurde und der hervorgehobenen Rahmenbox, erzeugt durch `\pd@tochighlight`. Voreingestellt ist: 0.5mm.

*option*  
`tocsecsep`

`tocsecsep`

Dieser Abstand ist vor einem Abschnitt eingefügt (ausgenommen, es ist das erste Element in dem Inhaltsverzeichnis). Voreingestellt ist: 2ex. Beachten Sie, dass wenn die Orientierung des Inhaltsverzeichnisses auf vertikal gesetzt ist, die Länge eine vertikale Auslassung erzeugt, anderenfalls erzeugt es eine horizontale Auslassung.

*option*  
`tocslidesep`

`tocslidesep`

Der Abstand ist vor anderen Eingaben eingefügt (ausgenommen es ist das erste Element in dem Inhaltsverzeichnis). Voreingestellt ist: 0ex. Wie bei `tocsecsep` ist der Effekt der Länge abhängig von der Ausrichtung des Inhaltsverzeichnisses.

*option*  
`tocsecindent`

`tocsecindent`

Ein horizontales Leerfeld links von der Abschnitteingabe. Voreingestellt ist: Opt.

*option*  
`tocslideindent`

`tocslideindent`

Ein horizontales Leerfeld links von der Folieneingabe. Die horizontale Auslassung wird nicht links von der Folieneingabe eingefügt, die vor dem ersten Abschnitt erscheint. Voreingestellt ist: Opt.

<i>option</i> tocsecm	<code>tocsecm</code>	Dies wird vor dem Schriftsetzen eines Abschnitts eingefügt. Es kann zum Markieren eines Abschnitts verwendet werden, zum Beispiel mit einer Linie im default-Stil. Voreingestellt ist: empty.
<i>option</i> toctcolor	<code>toctcolor</code>	Dies ist die Schriftfarbe, benutzt für nicht hervorgehobene Elemente im Inhaltsverzeichnis. Voreingestellt ist: black.
<i>option</i> tochltcolor	<code>tochltcolor</code>	Dies ist die Schriftfarbe, benutzbar für hervorgehobene Elemente im Inhaltsverzeichnis. Voreingestellt ist: white.
<i>option</i> tochlcolor	<code>tochlcolor</code>	Die ist die Farbe, die für den Rahmen hinter den hervorgehobenen Elementen benutzt wird. Voreingestellt ist: black.

## 9.7 Sonstige Optionen

Es gibt einige Optionen, die aus den Rahmen der vorherigen Abschnitte fallen. Diese werden im Folgenden besprochen.

<i>option</i> iacolor	<code>iacolor</code>	Die Option <code>iacolor</code> können Sie benutzen, um die Farbe, die für inaktive Symbole genutzt wird, zu spezifizieren. Sie wird beispielsweise von <code>\onslide</code> , <code>\pause</code> (siehe Abschnitt 4) und <code>\tableofcontents</code> (siehe Abschnitt 5.2) benutzt. Wenn <code>xcolor</code> von <code>powerdot</code> verwendet wird, können Sie hierbei spezielle Darstellungsarten wählen, wie
--------------------------	----------------------	--

```
iacolor=black!20
```

Der voreingestellte Wert für diesen Schlüssel ist `lightgray`.

Die folgenden Optionen steuern die Digitaluhr (siehe Abschnitt 2.1). Die Uhr ist ein gestaltbares Textfeld mit einem dynamischen Inhalt, was durch Javascript über `hyperref` Textfelder gesteuert wird. Einige Optionen für die Uhr arbeiten ähnlich wie andere, wie zum Beispiel beim Titelbaustein, aber es gibt ebenfalls spezielle Optionen.

<i>options</i> clockhook clockpos	<code>clockhook</code> <code>clockpos</code>	Diese arbeiten auf die gleiche Weise wie die <code>-hook-</code> und <code>-pos-</code> Bestandteile, die in Abschnitt 9.5 diskutiert wurden. Der voreingestellte Wert der <code>clockhook</code> ist <code>tr</code> .
<i>options</i> clockwidth clockheight	<code>clockwidth</code> <code>clockheight</code>	Diese steuern die Breite und die Höhe des Textfeldes, das die Uhr beinhaltet. Die voreingestellten Werte kommen von <code>hyperref</code> und haben ein Maß von <code>3cm</code> beziehungsweise von <code>\baselineskip</code> .
<i>option</i> clockcharsize	<code>clockcharsize</code>	Die Größe der Ziffern auf der Uhr. Voreingestellt ist <code>14pt</code> .
<i>option</i> clockalign	<code>clockalign</code>	Die Ausrichtung der Uhr im Textfeld. <code>0</code> bedeutet links ausgerichtet, <code>1</code> ist zentriert und <code>2</code> ist rechts ausgerichtet. Voreingestellt ist <code>2</code> .

*option*  
clockcolor

clockcolor

Dies legt die Schriftfarbe der Uhr fest. Der Wert muss eine bestimmte Farbe sein. Der voreingestellte Wert ist black.

## 9.8 Voreingestellte Templates

Nachfolgenden werden die voreingestellten Einstellungen der Schlüssel beschrieben. Diese können benutzt werden, wenn Sie keinen anderen Input für diese Schlüssel in eine bestimmte Schablone liefern. Wenn die voreingestellten Werte Ihre Bedürfnisse erfüllen, brauchen Sie diese nicht noch einmal in ihrem eigenen Stil spezifizieren.

```
titlehook=Bl,titlepos=,titlewidth=\slidewidth,
titlefont=\raggedright,texthook=tl,textpos=,
textwidth=\slidewidth,textfont=\raggedright,
textheight=\slideheight,
tochhook=tl,tocpos=,tocwidth=.2\slidewidth,
tocfont=\tiny\raggedright,
stochhook=tl,stocpos=,stocwidth=.2\slidewidth,
stocfont=\tiny\raggedright,
ntochhook=tl,ntocpos=,ntocwidth=.2\slidewidth,
ntocfont=\tiny\raggedright,
tocorient=v,stocorient=v,ntocorient=v,
tocfrsep=.5mm,tocsecsep=2ex,tocslidesep=0ex,
tocsecm=,toctcolor=black,tochlcolor=black,tochltcolor=white,
tocsecindent=0pt,tocslideindent=0pt,
lfhook=Bl,lfpos=,lffont=\scriptsize,lftemp=\pd@lf,
cfhook=B,cfpos=,cffont=\scriptsize,cftemp=\pd@cf,
rfhook=Br,rfpos=,rffont=\scriptsize,rftemp=\pd@rf\ifx\pd@rf
\@empty\else\ifx\theslide\@empty\else\ -- \fi\fi\theslide,
iacolor=lightgray,
clockhook=tr,clockpos=,clockwidth=3cm,clockheight=\baselineskip,
clockcharsize=14pt,clockalign=2,clockcolor=black
```

## 9.9 Der Hintergrund

Nur ein Argument des Makros `\pdefinemplate` ist noch unbesprochen. Dies ist das Befehls-`<commands>`-Argument. Dieses Argument kann jeden Code einbinden, den Sie ausführen möchten, *nachdem* die Optionen gesetzt wurden und *bevor* die Folienbausteine wie der Folientitel, Haupttext und Fußbereiche erstellt wurden. Dieses Argument ist konstruiert, um Deklarationen einzubinden, die den Hintergrund einer Folie erstellen und zum Beispiel `pstricks` benutzen. Aber es kann auch andere Befehle enthalten, die Sie zum Erstellen Ihrer Schablone brauchen.

Es ist wichtig festzuhalten, dass diese Befehle nicht unbedingt  $\TeX$ -Material kreieren, das Ihre Konstruktion der Folie zerstören könnte. Falls Sie das Wort „Hallo“ in der unteren linken Ecke der Folie platzieren möchten, schreiben Sie nicht „Hallo“, legen Sie aber die Breite, Höhe und Tiefe gleich der Null, zum Beispiel mit der Benutzung von `pstricks' \rput`.

```
\rput [bl] (0,0){Hello}
```

## 9.10 Titelfolie, Titel und Abschnitte

Wie zuvor erwähnt, muss der Stil, an dem Sie schreiben, definiert werden und damit zuletzt auch die Schablonen `slide` und `titleslide`. Letztere behandelt einige der Schlüssel in einer speziellen Weise.

Die Titelfolie (erzeugt durch `\maketitle`) setzt den Titel mit den Autor/en und das Datum in die Haupttextbox. Dies bedeutet, dass Sie eine Position für die Haupttextbox

(textpos) liefern müssen. Die Haupttextschriftgröße (zusammen mit den Erklärungen in dem textfont Schlüssel) wird für Autor/en und Datum benutzt. Die Erklärung wird titlefont für den Titel der Präsentation gebraucht. Dadurch formen Titel und Autor/en einen zusammenhängenden Block und es wird sichergestellt, dass lange Titel Autor/en nach unten verschieben, anstatt ihn/sie zu überschreiben

`\pd@slidetitle` Das Makro `\pd@slidetitle` wird verwendet, um den Folientitel auf die Folien zu setzen. Dieses Makro ist zum Beispiel mit `\pd@tocslide` vergleichbar. Es nimmt ein Argument an, das den Folientitel mit der richtigen Schriftart und Formation hat. Standardmäßig passt dieses Makro den Inhalt für das Schriftsetzen an, aber Sie können dieses Makro umdefinieren und somit seinen früheren Input erstellen, um die Schrift zu setzen. Ein Beispiel ist der fyma Stil, der den Titel unterstreicht, nachdem er in eine minipage gesetzt wurde und die den Mehrfachlinientitel unterstützt. Diese Makros haben Ähnlichkeit zu `\pd@slidetitle` und setzen den Titel auf die Titelfolie bzw. den Titel auf die Abschnittsfolien. Standardmäßig bestehen diese ebenso aus ihrem Argument (was der Titel der Präsentation oder der Titel eines Abschnitts ist). Aber dies kann auch umdefiniert werden, um so den früheren Input zu erstellen und so die Schrift zu setzen, wie bei `\pd@slidetitle`. Der Befehl `\section` benutzt (standardmäßig) die `slide`-Umgebung und setzt den Abschnittstitel in die Titelbox mit der Schriftart `titlefont`. Wenn Sie zum Beispiel diesen Standard ändern möchten und die `slide`-Umgebung, die `sectionslide`-Umgebung oder eine beliebige eigens erstellte Abschnittschablone auch für die Abschnitte nutzen möchten, ändern Sie die voreingestellte Schablone in Ihrem Stil mit

```
\setkeys [pd] {section} {sectemp=sectionslide}
```

Dies bedeutet, dass bei der Forderung des Benutzers nach `template=slide` in dem Befehl `\section` die `sectionslide`-Umgebung stillschweigend benutzt wird. Um Überraschungen zu vermeiden, sollte `sectionslide` vorzugsweise auf der `slide`-Umgebung basieren.

Eine ähnliche Option ist verfügbar in dem Fall, dass der Benutzer `template=wideslide` fordert. Beispielsweise diese:

```
\setkeys [pd] {section} {widesectemp=sectionwideslide}
```

Jedes Mal wenn der Benutzer ein `wideslide` anfordert, gebraucht für `\section`, wird stattdessen die `sectionwideslide`-Umgebung benutzt. Bei anderen Inputs für den `template`-Schlüssel erfolgt keine spezielle Bearbeitung.

Beachten Sie, dass diese Schlüssel in der `section` Schlüssel Familie verfügbar sind und dass Sie diese nicht für den Befehl `\pddefinestyle` verwenden können.

## 9.11 Das Testen des Stils

powerdot hat eine Testdatei, die die meisten Stile testet. Die Testdatei kann angefertigt werden, indem  $\LaTeX$  mit `powerdot.dtx` durchläuft. Diese generiert `powerdot-styletest.tex`, was Ihnen hilft die Eingaben zu kontrollieren. Wenden Sie sich an uns, wenn Sie Ihren Stil `powerdot` beisteuern möchten. Siehe ebenfalls Abschnitt 11.

## 10 Die Benutzung von $\LaTeX$ für Präsentationen

$\LaTeX$  [6] ist ein WYSIWYM (What You See Is What You Mean) Dokumentprozessor, basierend auf  $\LaTeX$ . Es unterstützt  $\LaTeX$ -Standardklassen, braucht aber spezielle Dateien, genannt Layout-Dateien, um nicht standardisierte Klassen, wie `powerdot`, zu unterstützen.

Um  $\LaTeX$  für `powerdot`-Präsentationen zu verwenden, kopieren Sie die Layout-Datei `powerdot.layout` in das  $\LaTeX$ -Layout-Datenverzeichnis. Diese Datei finden Sie in Ihrem  $\LaTeX$ -Installationsverzeichnis (bei Linuxsystemen) unter dem Pfad `texmf/doc/`

latex/powerdot. Falls Sie ihn hier nicht finden, können Sie ihn auch von CTAN herunterladen CTAN:/macros/latex/contrib/powerdot. Sobald dies getan ist, konfigurieren Sie LyX (Edit > Reconfigure und und starten Sie LyX danach neu). Jetzt können Sie die powerdot-Dokumentenklasse wie eine beliebige andere unterstützte Klasse benutzen. Wählen Sie powerdot presentation als Dokumentklasse aus. Für mehr Informationen schauen Sie in die LyX-Dokumentation, die unter dem Help-Menü abrufbar ist.

## 10.1 Wie das Layout benutzt wird

Das powerdot-LyX-Layout bietet einige Umgebungen<sup>12</sup>, die in LyX verwendet werden können. Manche dieser Umgebungen (beispielsweise Title oder Itemize) sind normal nutzbar, seit sie ebenfalls in Standarddokumentenklassen wie article existieren. Mehr Informationen über die Standardumgebungen sind in der LyX-Dokumentation zu finden.

Dieser Abschnitt will Ihnen erklären, wie die speziellen powerdot-Umgebungen Slide, WideSlide, EmptySlide und Note zu benutzen sind. Diese Umgebungen entsprechen den powerdot-Umgebungen slide, emptyslide, wideslide und note.

Beginnen wird mit einem einfachen Beispiel. Der folgende  $\text{\LaTeX}$ -Code

```
\begin{slide}{Slide title}
Slide content.
\end{slide}
```

ist bei der Benutzung der folgenden LyX-Umgebungen erhältlich. Die rechte Spalte repräsentiert den Text, eingegeben in das LyX-Fenster und die linke Spalte repräsentiert die Umgebung, angewandt auf diesen Text.

Slide	Slide title
Standard	Slide content.
EndSlide	

Einige Anmerkungen, bezogen auf dieses Beispiel.

- Sie können dieses Umgebungsmenü (unter dem Menübalken, oberere linke Ecke) verwenden, um die Umgebung zu wechseln, angewandt auf den Text.
- Der Folientitel sollte in die Zeile der Slide-Umgebung geschrieben werden.
- EndSlide beendet die Folie und lässt die Linie unbeschrieben.

In dem LyX-Fenster wird die Slide-Umgebung (der Folientitel) in magenta ausgegeben, der WideSlide-Stil in grün, der EmptySlide-Stil in cyan und der Note-Stil in rot und daher sind diese leicht identifizierbar.

Hier ist ein anderes Beispiel.

```
\begin{slide}{First slide title}
The first slide.
\end{slide}
\begin{note}{First note title}
The first note, concerning slide 1.
\end{note}
\begin{slide}{Second slide title}
The second slide.
\end{slide}
```

Das lässt sich mit LyX folgendermaßen erstellen.

Slide	First slide title
Standard	The first slide.
Note	First note title
Standard	The first note, concerning slide 1.

<sup>12</sup>Nicht mit den  $\text{\LaTeX}$ -Umgebungen verwechseln.

```
Slide      Second slide title
Standard   The second slide.
EndSlide
```

Dieses Beispiel demonstriert, dass es oft genügt, den `EndSlide`-Stil nach der letzten Folie oder Notiz einzufügen. Nur wenn Sie bestimmtes Material nicht als Teil einer Folie wollen, müssen Sie die vorausgehende Folie manuell mit dem `EndSlide`-Stil beenden. Beispiel:

```
Slide      First slide title
Standard   The first slide.
EndSlide
[ERT box with some material]
Slide      Second slide title
...
```

Optionen können für Folienumgebungen mit der Benutzung von `Insert > Short title` vor dem Folientitel übermittelt werden. Das folgende Beispiel gebraucht die direkte `direct` Methode (siehe Abschnitt 6.4) im Kurztitel-Argument (begrenzt durch einen eckigen Bereich), um eine `lstlisting`-Umgebung (definiert vom dem `listings` Packet) binnen des Folieninhalts zu erstellen.

```
Slide      [method=direct]Example of LaTeX source code
Standard   Here's the \HelloWorld command:
[ERT box:
 \lstset{language=[LaTeX]TeX}
 \begin{lstlisting}
 \newcommand{\HelloWorld}{Hello World!}
 \end{lstlisting}
]
EndSlide
```

Beachten Sie, dass Sie nicht verpflichtet sind, eine `verbatim`-Umgebung zu nutzen, um `\HelloWorld` in das `LyX`-Fenster zu schreiben, weil `LyX` direkt einen wörtlichen Standard unterstützt.<sup>13</sup> Folglich ist die Benutzung der Methoden der Folienaufbereitung `direct` und `file` nicht notwendig, wenn Sie einen wörtlichen Standard gebrauchen, aber es ist notwendig, wenn Sie fortgeschrittene Dinge tun möchten, wie im obigen Beispiel.

## 10.2 Unterstützung der Syntax

Dieser Abschnitt listet Optionen, Befehle und Umgebungen auf, die durch das `LyX`-Interface direkt unterstützt werden, ohne eine ERT-Box zu benutzen (`TEX`-mode).

Alle Optionsklassen (siehe Abschnitt 2.1) werden durch den `Layout > Document-Dialog` unterstützt (`Layout`-Tafel). Optionen für den Befehl `\pdsetup` (siehe Abschnitt 2) sollten in dem `Präambel-Preamble`-Teilabschnitt des `Layout > Document-Dialogs` spezifiziert werden.

Die Tabelle 3 listet die `powerdot`-Befehle auf, die die von `LyX` unterstützt werden. Die Tabelle 4 listet die `powerdot`Umgebungen auf, die zusätzlich zu den vorher diskutierten `slide`-, `wideslide`-, `note`- und `emptyslide`-Umgebungen auch von `LyX`unterstützt werden. Tabelle 5 listet Befehle auf, die nur bei Benutzung der ERT-Box (durch `Insert > TeX`) durchgeführt werden können.

Beachten Sie, dass Sie die Ablage gebrauchen dürfen, um häufige Befehle, wie `\pause`. Schließlich listet die Tabelle Finally 6 zusätzliche Befehle und Umgebungen auf, die vom `Layout` unterstützt werden.

<sup>13</sup>`LyX` übersetzt spezielle Zeichen in ihren dazugehörigen  $\LaTeX$ -Befehl. Zum Beispiel das `backslash`-Zeichen wird in `\textbackslash` übersetzt. Resultierend ist die Schriftart nicht die gleiche wie in der wahren wortgetreuen Übersetzung und Sie könnten dies durch den `Layout > Character`-Dialog ändern.

Befehl	Methode in L <sup>A</sup> T <sub>E</sub> X
<code>\title</code>	Benutzt Title-Umgebung.
<code>\author</code>	Benutzt Author-Umgebung.
<code>\date</code>	Benutzt Date-Umgebung.
<code>\maketitle</code>	Direkt von L <sup>A</sup> T <sub>E</sub> X ausgeführt.
<code>\section</code>	Benutzt die Section-Umgebung. Optionen für diesen Befehl (siehe Abschnitt 5.1) können durch <code>Insert▷ Short title</code> vor dem Abschnittstitel spezifiziert werden.
<code>\tableofcontents</code>	Benutzt <code>Insert▷ Lists &amp; TOC▷ Table of contents</code> . und braucht eine ERT-Box, wenn ein optionales Argument benutzt werden soll. Siehe unten.

Tabelle 3: Unterstützte powerdot-Befehle von L<sup>A</sup>T<sub>E</sub>X

Umgebung	Methode von L <sup>A</sup> T <sub>E</sub> X
<code>itemize</code>	Benutzt <code>Itemize-</code> und <code>ItemizeType1-</code> Umgebungen. Letzteres wird eine Liste mit <code>type=1</code> (sehen Sie Abschnitt 4.2) erstellen.
<code>enumerate</code>	Benutzt <code>Enumerate-</code> und <code>EnumerateType1-</code> Umgebungen.
<code>thebibliography</code>	Benutzt <code>Bibliography-</code> Umgebung.

Tabelle 4: Unterstützte powerdot-Umgebungen von L<sup>A</sup>T<sub>E</sub>X

### 10.3 Programmübersetzung mit L<sup>A</sup>T<sub>E</sub>X

Zuerst stellen Sie sicher, dass Sie auch den Abschnitt 8 gelesen haben. Dann, um ein einwandfreies PostScript-Dokument oder eine PDF-Datei zu bekommen, müssen Sie Ihre L<sup>A</sup>T<sub>E</sub>X-Dokumentbestandteile, abhängig davon welches Papier und welche Ausrichtung Sie bevorzugen, anpassen und auswählen. Wenn das L<sup>A</sup>T<sub>E</sub>X geöffnet ist, gehen Sie zu `Layout▷ Document-Dialog`. In dem `Layout-Ausschnitt` setzen Sie die `nopsheader-`, `orient-` und `paper-`Schlüssel als Klassenoptionen (siehe Abschnitt 2.1 mit einer Beschreibung). Gehen Sie dann zum `Paper-Ausschnitt` und wählen Sie die dementsprechende Papiergröße und Ausrichtung (Sie können die `letter-`Papiergröße auswählen, wenn Sie `paper=screen` in der Klassenoption setzen). Schließlich gehen Sie zum `View-` (oder `File▷ Export-`)-Menü und wählen Sie Ihren Output (PostScript oder PDF).

### 10.4 Erweiterung des Layouts

Wenn Sie einen individuell gefertigten Stil (siehe Abschnitt 9), welcher individuelle Schablonen definiert, erstellt haben, müssen Sie die `Layout-Datei`<sup>14</sup> erweitern, sodass diese Schablonen ebenfalls von L<sup>A</sup>T<sub>E</sub>X unterstützt werden. Die Erklärung unten unterstellt, dass Sie eine Schablone (genannt `sunnyslide`) definiert haben. Um diese neue Schablone mit L<sup>A</sup>T<sub>E</sub>X zu unterstützen, müssen Sie den folgenden Befehl benutzen.

```
\pddefinelyxtemplate<cs>{<template>}
```

`\pddefinelyxtemplate`

Dies wird die Befehlssequenz `<cs>` derart definieren, dass Sie eine Folie mit der Schablone `<template>` erzeugen können (die mit der Benutzung von `\pddefinemplate`

<sup>14</sup>Die LPPL schreibt vor, eine Datei umzubennen, wenn Sie diese modifizieren, um Verwechslungen zu vermeiden.

Befehl	Methode bei LyX
<code>\and</code>	innerhalb der Author-Umgebung.
<code>\pause</code>	
<code>\item</code>	Eine ERT-Box ist nur erforderlich für das optionale Argument, nicht obligatorisch für Overlay-Spezifikationen.
<code>\onslide</code>	Und die Versionen <code>\onslide+</code> und <code>\onslide*</code> .
<code>\twocolumn</code>	
<code>\tableofcontents</code>	Nur bei der Benutzung eines optionalen Arguments.

Tabelle 5: powerdot-Befehle, die eine ERT-Box in LyX benötigen

Umg./Befehl	Methode bei LyX
<code>quote</code>	Benutzt <code>quote</code> -Umgebung.
<code>quotation</code>	Benutzt <code>quotation</code> -Umgebung.
<code>verse</code>	Benutzt <code>verse</code> -Umgebung <code>environment</code> .
<code>\caption</code>	Benutzt <code>caption</code> -Umgebung innerhalb einer Standardpufferumgebung.

Tabelle 6: Zusätzliche Umgebungen für LyX

definiert wurde). Diese neue Befehlssequenz kann wie folgt in der Layout-Datei gebraucht werden.

```
# SunnySlide environment definition
Style SunnySlide
  CopyStyle Slide
  LatexName lyxend\lyxsunnyslide
  Font
    Color Yellow
  EndFont
  Preamble
    \pddefinelyxtemplate\lyxsunnyslide{sunnyslide}
  EndPreamble
End
```

Beachten Sie, dass das `LatexName`-Feld mit `lyxend` beginnen muss. Die Definition der LyX-Schablone wurde zwischen `Preamble` und `EndPreamble` eingefügt, was gewährleistet, dass die neue LyX-Umgebung in jeder Präsentation funktionieren wird. Nachdem die Layout-Datei identifiziert wurde, vergessen Sie nicht LyX wieder zu starten. Für mehr Informationen über das Erstellen einer LyX-Umgebung schauen Sie in die Dokumentation für LyX im Hilfe-HelpMenü.

## 11 Fragen

### 11.1 Häufig gestellte Fragen

Dieser Abschnitt ist häufig gestellten Fragen gewidmet. Lesen Sie ihn aufmerksam, Ihre Probleme könnten in diesem Abschnitt gelöst werden.

**Q1** Hat powerdot Beispieldateien? Wo kann ich diese finden?

**A1** powerdot kommt mit einigen Beispiele, die sich in ihrem Pfad bei der  $\text{\LaTeX}$ -Installation befinden. Genauer: `texmf/doc/latex/powerdot`. Wenn Sie diese hier nicht finden können, laden Sie sie von CTAN herunter: `CTAN:/macros/latex/contrib/powerdot` [7].

**Q2** Ich bekomme Fehler oder unerwartete Outputs, wenn die simpelste Programmübersetzung läuft!

**A2** Haben Sie Abschnitt 8 gelesen?

**Q3** Ich habe einen Tippfehler in den Foliencode gemacht, ließ die Datei durchlaufen, berichtigte den Tippfehler und ließ die Datei erneut durchlaufen. Aber nun bekam ich einen Fehler, der sich nicht entfernen ließ.

**A3** Entfernen Sie die `.bm-` und `.toc-`Dateien und versuchen Sie es erneut.

**Q4** `\pause` funktioniert nicht in der `align`-Umgebung<sup>15</sup>

**A4** `align` unternimmt einige komplizierte Dinge, die es unmöglichen machen, `\pause` zu verwenden. Benutzen Sie stattdessen `\onslide`, siehe Abschnitt 4.4.

**Q5** Meine `pstricks`-Knoten treten auf allen Overlays auf. Außerdem: `color` scheint nicht mit `\onslide` zu funktionieren.

**A5** Einige PostScript-Tricks wie `nodes` und `color` funktionieren nicht mit `\onslide`. Benutzen Sie stattdessen `\onslide*` Betrachten Sie das folgende Beispiel.

```
\documentclass{powerdot}
\usepackage{pst-node}
\begin{document}
\begin{slide}{Color}
\onslide*{2}{\cnode(0,-5pt){2pt}{A}}
This is {\onslide*{2-}{\color{red}} red} text.
\onslide*{2}{\cnode(0,-5pt){2pt}{B}}
\onslide{2}{\ncline{A}{B}}
\end{slide}
\end{document}
```

**Q6** Muss ich Stil-Dateien editieren, um diese ein wenig zu verändern?

**A6** Nein, Sie müssen keine Stildateien editieren. Sie können jeden Teil eines bestimmten Stils verändern, indem Sie die Befehle `\pddefinemplate` und `\pddefinepalettes` verwenden. Hier ist ein Beispiel, das den rechten Fuß aus dem `default`-Stil entfernt, den linken Fuß in das Zentrum verschiebt und eine andere Farbskala hinzufügt.

```
\documentclass{powerdot}
\pddefinemplate[slide]{slide}{
  lfpos=,rfpos=,cftemp=\theslide
}{}
\pddefinepalettes{mypalette}{
  \definecolor{pdcolor1}{rgb}{.27,.31,.44}
  \definecolor{pdcolor2}{rgb}{.85,.85,.92}
  \definecolor{pdcolor3}{rgb}{.8,.75,.98}
}
\pdsetup{palette=mypalette}
\begin{document}
\begin{slide}{Title}
\end{slide}
\end{document}
```

Betrachten Sie den Abschnitt 9 für mehr Informationen über diese beiden Befehle.

**Q7** Kann ich bei diesem Projekt mitarbeiten?

**A7** Sicher. Wenn Sie einen Fehler<sup>16</sup> oder Tippfehler finden, senden Sie uns eine Nachricht über die Verteilerliste (siehe Abschnitt 11.2). Wenn Sie einen eigenen Stil erstellt haben, der sich von den existierenden Stilen unterscheidet und Sie diesen in `powerdot` aufgenommen sehen wollen, informieren Sie uns bitte durch eine private E-Mail und wir werden Ihren Beitrag prüfen. Beachten Sie, dass aufgenommene Beiträge unter das allgemeine `powerdot` Lizenz und Urheberrecht fallen. Aber Ihr Name wird in die

<sup>15</sup>Es gibt einige andere Umgebungen, die ähnlich funktionieren wie `align`. Ein weiteres Beispiel ist die `split`-Umgebung, aber dies (oft in dem `amsmath`-Paket) kann die gleichen Probleme mit `\pause` verursachen.

<sup>16</sup>Stellen Sie sicher, dass Sie bestätigen können, dass der Fehler wirklich von `powerdot` verursacht wird und nicht von einem anderen Paket, das Sie benutzen.

Dokumentation aufgenommen, wenn Sie einen Beitrag leisten. Dies wird getan, um zu garantieren, dass die Dateien angepasst werden, falls Wartungsarbeiten nötig sind.

Wenn Ihre Fragen an diesem Punkt nicht beantwortet wurden, gehen Sie zum nächsten Abschnitt, um zu Lesen, wo Sie mehr Antworten finden.

## 11.2 Mailingliste

powerdot hat eine Mailing-Liste bei `freelists.org` und hat die Website hier:

`http://www.freelists.org/list/powerdot`

Da gibt es einen Link zur Archivliste ('List Archive'). Bitte durchsuchen Sie das Archiv, bevor Sie ihre Frage stellen. Ihr Problem wurde vielleicht schon in der Vergangenheit gelöst.

Wenn dies nicht der Fall ist, benutzen Sie die Box auf der Seite, um Ihre E-Mail-Adresse einzufügen; wählen Sie die Aktion 'Subscribe' und klicken Sie auf 'Go!'. Folgen Sie dann den Anweisungen, die Sie per E-Mail erreicht haben. An diesem bestimmten Zeitpunkt können Sie sich zum ersten Mal mit einem autorisierten Code, der Ihnen per E-Mail zugesandt wurde, einloggen. Nachdem Sie sich eingeloggt haben, können Sie sich ein eigenes Passwort für zukünftige Arbeitssitzungen unter dem Button 'Hauptmenü' erstellen. Die anderen Button versorgen Sie mit inigen Informationen und Optionen für Ihren Account.

Wenn Sie alles eingestellt haben, können Sie der Liste schreiben, indem Sie eine E-Mail an folgende Adresse senden:

`powerdot[at]freelists[dot]org`

Wenn Sie der Liste schreiben, behalten Sie die folgenden Kernpunkte im Hinterkopf.

1. Wir sind Freiwillige!
2. Beziehen Sie Ihre Fragen auf powerdot.
3. Liefern Sie ein *minimales* Beispiel, was Ihr Problem deutlich macht.
4. Senden Sie an die Liste keine großen Dateien.

Wir hoffen, dass Sie Gefallen an diesem Service finden.

## 12 Quelltextdokumentation

Für den Fall, dass Sie die Paketdateien von der Quelle aus erneuern möchten oder Sie einen Blick auf die Quelltextbeschreibung werfen möchten, suchen Sie nach `powerdot.dtx` und darin nach `\OnlyDescription` und entfernen Sie dies und führen sie nachfolgende Befehle aus (bei der aktuellen Version ist es nicht möglich, die `powerdot.dtx` auf diesem Weg zu modifizieren bzw. eine neue `.sty` Datei zu erzeugen. C.R.):

```
latex powerdot.tex
latex powerdot.tex
bibtex powerdot
makeindex -s gglo.ist -o powerdot.gls powerdot.glo
makeindex -s gind.ist -o powerdot.ind powerdot.idx
latex powerdot.tex
latex powerdot.tex
```

## 13 Implementierung

### 13.1 Allgemeiner Aufbau

Dieses Kapitel erklärt die allgemeine Vorstellung zu dieser Klasse, wie die Seitenabmessungen gewählt sind und wie Folien erstellt werden. Begonnen wird mit dem Referat.

Diese Klasse benutzt die gleiche Idee wie die Pakete `prosper` und `HA-prosper`, nämlich, dass wir einen Hintergrund mit `pstricks` erzeugen und zur Textpositionierung `\rput` und einige `minipage`-Umgebungen verwenden. Dies ist einfacher, als eine eigene Ausgaberroutine zu schreiben, die das ganze Material verarbeiten kann, sie bricht aber die Seiten nicht um, wenn wir es nicht wollen. Wenn wir die Beschreibung dieser Klasse beginnen, untersuchen wir zuerst, wie wir mit `landscape`-Folien und Papierformaten umgehen.

`landscape`-Folien erzeugen, kann man mit `geometry`. Das Papierformat war ein größeres Problem. Da  $\TeX$  nicht so große Schriftschnitte hat, wie wir sie für Präsentationen brauchen, mussten wir die üblichen Schriften irgendwie skalieren. Dies könnte durch die Verwendung ungewöhnlicher Papierformate geschehen (wie in `beamer` geschehen) und daraus weitergeben an `ps2pdf` zum Ausschneiden des gesamten redundanten Materials. Dies führt aber nicht zur Erzeugung von benutzbaren DVI- oder PS-Dateien. Stattdessen entschieden wir uns, die DVI-Vergrößerung zu nutzen. Die gesamte Folie ist um den Faktor 2 vergrößert und nur das obere linke Quadrat des Papiers wird verwendet (dies kann man jedoch nicht aufgrund der Vergrößerung bemerken). Als wir beschlossen hatten, das gesamte Material in `minipages` mit `\rput` zu platzieren, benötigten wir die Seite nicht mehr und es war einfacher, `geometry` zu nehmen, um die Seitenränder zu entfernen. Insgesamt so, dass die linke untere Ecke  $(0, 0)$  wurde und die obere rechte Ecke  $(\backslash\slidewidth, \backslash\slideheight)$ .

Eine zusätzliche Bemerkung ist zu dem Papiertyp notwendig. Wir fanden, dass die meisten Konfigurationen von `dvips` mit dem `A4size`-Code bei den A4-Formaten zu finden sind. Dieser Code schreibt nicht explizit einen PostScript-Papier-Befehl in das PS-Dokument und deshalb wissen Programme, die diese PS nehmen (wie `ps2pdf`), nicht, welches Papierformat zu nehmen ist und kommen auf den Standard zurück, in den meisten Fällen auf das `letter`-Format. Dies ist natürlich etwas, was wir nicht wollen und daher schreibt `powerdot` selbst diese Befehle in die PS, um eine einwandfreie Verarbeitung der PS zu garantieren.

Nun, nachdem wir entschieden haben, wie die richtige DVI-, PS- und PDF-Ausgabe zu erstellen ist, bestand die nächste Aufgabe darin, eine einfache Schnittstelle für die Overlays zu schaffen. Um zu vermeiden, die Overlays zu zählen, so wie es mit `prosper` nötig ist, implementierten wir ein System, das zuerst die gesamten Körper in einer Umgebung eines Makros sammelt, die zu verschiedenen Zeitpunkten wieder verwendet werden kann. Während der Ausführung des Körpers erhalten die Overlay-Befehle, wie `\onslide`, eine Aufzeichnung der größten Zahl, die sie finden, und das ist die Zahl der zu produzierenden Overlays. Bekommend den Körper der Umgebung, wird das Material bis zu nächsten Auftreten von `\end` gesammelt. Wenn die Kontrollsequenz das richtige Argument hat (nämlich die Anfangsfolie), dann stoppen wir das Scannen und beginnen mit der Folienverarbeitung. Bei normalen Verlauf werden Sie nichts bemerken, aber mit versteckten Spezialtricks, wie `\end{slide}`, in einem Makro wird der Prozess auf der Suche nach `\end{slide}` scheitern, der `input stream` wird nicht erweitert. Oft kann ein `work around` gefunden werden. Werfen Sie beispielsweise einen Blick darauf, wie `\section` eine Folie erzeugt.

Die nächste Aufgabe bestand darin, einen einfachen Weg zu entwickeln, Vorlagen zu erstellen. Die Klasse basiert auf der Idee, einen Hintergrund für die gesamte Materialverteilung zu haben; das Vorlagensystem folgt dieser Idee. Ein Argument kann verwendet werden, um den Hintergrund zu erzeugen, das andere Argument kontrolliert via Schlüssel und Optionen verschiedene Eigenschaften des Materials, das plat-

ziert werden soll. Durch das Hinzufügen eines Schlüssels `ifsetup` konnte die volle Kontrolle über die Gestaltung der Vorlage in allen möglichen Einrichtungen, die der Benutzer auswählt, gewonnen werden.

Die abschließende Aufgabe für die Klasse war es, sie in allen ‘Details’ auszufüllen. Alle Mechanismen waren vorhanden, aber manchmal sollten sie nicht aktiv sein. Beispielsweise sollten Overlays nicht im Handout-Modus erstellt werden. Andere Dinge, die zur Klasse hinzugefügt wurden, schützten die Zähler der Overlays, fassten das Layout der Foliennummer, von Fußnoten und der Bibliografieumgebung, leeren Folien, etcetera an.

Wir hoffen, dass dieser Abschnitt etwas erklärt hat, was Sie beim Lesen des nächsten Abschnitts mit den Codes sehen werden und warum wir dies und auf diese Weise getan haben. (Codes sind in dieser Version der Dokumentation nicht mehr enthalten, siehe dafür die `powerdot`-Stildatei. C.R.)

## Literatur

- [1] Hendri Adriaens. HA-prosper package. CTAN:/macros/latex/contrib/HA-prosper.
- [2] Hendri Adriaens. xkeyval package. CTAN:/macros/latex/contrib/xkeyval.
- [3] Donald Arseneau. random.tex. CTAN:/macros/generic/misc.
- [4] Javier Bezos. enumitem package. CTAN:/macros/latex/contrib/enumitem.
- [5] David Carlisle. graphics bundle. CTAN:/macros/latex/required/graphics.
- [6] L<sup>A</sup>T<sub>E</sub>X crew. L<sup>A</sup>T<sub>E</sub>X website. <http://www.lyx.org>.
- [7] CTAN crew. The Comprehensive TeX Archive Network. <http://www.ctan.org>.
- [8] Patrick W. Daly. natbib package. CTAN:/macros/latex/contrib/natbib.
- [9] Frédéric Goualard and Peter Møller Neergaard. prosper class. CTAN:/macros/latex/contrib/prosper.
- [10] Uwe Kern. xcolor package. CTAN:/macros/latex/contrib/xcolor.
- [11] James Kilfiger and Wolfgang May. extsizes bundle. CTAN:/macros/latex/contrib/extsizes.
- [12] Frank Mittelbach and Michel Goossens. *The L<sup>A</sup>T<sub>E</sub>X Companion*. Tools and Techniques for Computer Typesetting. Addison-Wesley, Boston, Massachusetts, 2 edition, 2004. With Johannes Braams, David Carlisle, and Chris Rowley.
- [13] Adobe Solutions Network. Acrobat JavaScript Scripting Reference. <http://partners.adobe.com/public/developer/en/acrobat/sdk/pdf/javascript/AcroJS.pdf>.
- [14] Sebastian Rahtz and Heiko Overdiek. hyperref package. CTAN:/macros/latex/contrib/hyperref.
- [15] Hideo Umeki. geometry package. CTAN:/macros/latex/contrib/geometry.
- [16] Herbert Voß. PSTricks website. <http://tug.org/pstricks/>.
- [17] Timothy Van Zandt et al. PSTricks package, v1.07, 2005/05/06. CTAN:/graphics/pstricks.

## Danksagungen

Die Autoren danken Mael Hilléreau für das Beisteuern der L<sup>A</sup>T<sub>E</sub>X-Layoutdatei und deren Beschreibung. Ferner all jenen, die Stile für Powerdot bereitstellten (Abschnitt 7) und darüberhinaus allen, die an diesem Paket in der einen oder anderen Weise beteiligt waren.

Ramon van den Akker, Pavel Čížek, Darren Dale, Hans Marius Eikseth, Morten Høgholm, András Horváth, Laurent Jacques, Akira Kakuto, Uwe Kern, Kyanh, Theo Stewart, Don P. Story and Herbert Voß.

Wir hoffen, dass wir niemanden vergessen haben.

## Versiongeschichte

Mehr Informationen über bug fixes, typeset the source code Documentation (finden Sie in Abschnitt 12).

<b>v1.0</b>	<i>(2005/09/04)</i>
General: Initial release	1
<b>v1.1</b>	<i>(2005/09/19)</i>
General: <code>blackslide</code> options adds hyperlink to slide and section titles	1
Added <code>elcolors</code> , <code>aggie</code> , <code>husky</code> and <code>sailor</code> styles	1
Added <code>tocsecindent</code> and <code>tocslideindent</code> options	1
Added graphical examples of styles to documentation	1
Added L <sup>A</sup> T <sub>E</sub> X layout, description and example	1
Changed <code>size=10</code> to <code>size=10pt</code>	1
Extended FAQ	1
Fixed some small bugs	1
Improved <code>tycja</code> , <code>ciment</code> and <code>fyma</code> styles	1
Improved section title handling	1
<b>v1.2</b>	<i>(2005/10/09)</i>
General: Added <code>upen</code> and <code>bframe</code> styles	1
Simplified coding of most styles	1
Solved some small bugs	1
Speeded up compilations	1
<b>v1.3</b>	<i>(2005/12/06)</i>
General: Added <code>horatio</code> , <code>paintings</code> , <code>klope</code> , <code>jefka</code> and <code>pazik</code> styles	1
Added clock feature	1
Added examples and example file	1
Added logo feature	1
Added optional argument to <code>\maketitle</code>	1
Added palettes feature	1
Added palettes to default, <code>fyma</code> and <code>sailor</code> styles	1
Added possibility to create horizontal table of contents	1
Added random dots feature	1
Added two slide processing methods to do verbatim on slides easily	1
Cleaned up options	1
Improved <code>figure</code> and <code>table</code> handling	1
Revised docs	1
Solved some small bugs	1
Updated all styles	1
Updated L <sup>A</sup> T <sub>E</sub> X example and information	1
<b>v1.4</b>	<i>(2005/12/10)</i>
General: Added <code>cf</code> option	1
Moved <code>lf</code> and <code>rf</code> keys from <code>global</code> to <code>glslide</code> family	1
Moved footers out of slide box in handout mode	1
Solved some small bugs	1
Updated styles	1

<b>v1.4</b>	<i>(2008/08/24)</i>
General: blackslide replaced by the more general pauseslide	1
pauseslide does not get randomdots anymore	1
Added clockformat and clockrefresh keys to global family	1
Added option for no frames in handout mode	1
Renames nopagebreaks option to nohandoutpagebreaks	1
<b>v1.4a</b>	<i>(2010/10/07)</i>
General: fixed bug with geometry 5.x (hv)	1
<b>v1.4b</b>	<i>(2010/11/16)</i>
General: fixed bug with centered footnote (hv)	1
<b>v1.4c</b>	<i>(2010/11/16)</i>
General: fixed bug with missing macros for centered footer (hv)	1
<b>v1.4d</b>	<i>(2010/12/03)</i>
General: fixed misleading comment header of powerdot.cls (hv)	1
<b>v1.4e</b>	<i>(2010/12/18)</i>
General: fixed bugs with enumitem (hv)	1
<b>v1.4f</b>	<i>(2011/04/01)</i>
General: added paper size smartboard (hv)	1
<b>v1.4g</b>	<i>(2011/04/25)</i>
General: load enumitem version 2.2 and younger (hv)	1

## Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in **roman** refer to the code lines where the entry is used.

<b>Symbols</b>	
-font (option)	35
-height (option)	35
-hook (option)	34
-orient (option)	35
-pos (option)	34
-temp (option)	35
-width (option)	35
<b>A</b>	
\and	11
\author	11
<b>B</b>	
bm (option)	12
bottomsep (option)	22
<b>C</b>	
cf (option)	8
cf- (option)	34
clock (option)	6
clockalign (option)	37
clockcharsize (option)	37
clockcolor (option)	38
clockformat (option)	8
clockheight (option)	37
clockhook (option)	37
clockpos (option)	37
clockrefresh (option)	8
clockwidth (option)	37
colsep (option)	22
content (option)	19
counters (option)	7
<b>D</b>	
\date	11
dbright (option)	10
display (option)	5
dmaxdots (option)	10
dmaxheight (option)	10
dmaxsize (option)	10
dmaxwidth (option)	10
dmindots (option)	10
dminheight (option)	10
dminsize (option)	10
dminwidth (option)	10
dprop (option)	10
<b>E</b>	
emptyslide (environment)	20
enumerate (option)	8
<b>F</b>	
fleqn (option)	5
frsep (option)	22
<b>H</b>	
hlentries (option)	6
hlsections (option)	6
<b>I</b>	
iacolor (option)	37
ifsetup (option)	32

