DVI und DVILW
Grafikfähige DVI-Treiber für TEX

Gerhard Wilhelms Ingo Eichenseher Markus Zahn

Version 3.62, Februar 1994
gesetzt am 15. Februar 1994

Inhaltsverzeichnis

Einleitung — DVI-Treiber

DVI und DVILW sind Treiber für TEX-DVI-Dateien für den Atari ST, Apollo Workstations unter GPR, Workstations mit X11/Motif, den Commodore Amiga und den IBM PC, die sich durch besondere Grafikfähigkeiten, sparsame Speicherplatzbenutzung und Verwendung von virtuellen Zeichensätzen auszeichnen. Zudem sind die Treiber über (Klartext-)Parameterdateien einfach in bestehende TEX-Systeme zu integrieren.

DVI arbeitet wahlweise in einem Gem-Fenster des Atari ST, auf Amigas ab Amiga OS 2.04, in einem Bildschirm-PAD einer Apollo, in einem X11-Fenster, auf PCs mit gängiger Grafikkarte (CGA, EGA, VGA, Super-VGA, oder Hercules), sowie mit EPSON-kompatiblen 9-Nadel Druckern, NEC-kompatiblen 24-Nadel Druckern, Canon-BJ-300- oder HP-Laserjet-kompatiblen Laserdruckern. Eine Anpassung auf andere Computer ist problemlos möglich, falls große Speicherbereiche (ca. 0.5 MB) alloziert werden können und ein grafikfähiger Bildschirm zur Verfügung steht.

DVILW arbeitet mit PostScript-Laserdruckern und ist auf dem Atari ST, dem Commodore Amiga, diversen Workstations und dem IBM PC implementiert. Der PostScript-Prolog basiert auf dem Prolog des DVIALW-PostScript-Treibers aus der Treiberfamilie von Nelson Beebe, hat im Lauf der Zeit jedoch umfangreiche Änderungen erfahren. Der Programmcode ist eine vollständige Eigenentwicklung, da der originale DVIALW-Code sich als deutlich zu langsam erwies. Zwei Besonderheiten von DVILW verdienen besondere Erwähnung. Zum einen können die PC- und Atari-Versionen direkt — ohne Zwischendatei — mittels einer ISDN-Schnittstelle z.B. über eine Siemens Hicom Telefonanlage oder über die RS232-Schnittstelle drucken, zum anderen ist es durch Verwendung virtueller Zeichensätze möglich, an Stelle der TEX-Zeichensätze die PostScript-Zeichensätze zu verwenden.

Unseren besonderen Dank möchten wir an dieser Stelle folgenden Personen aussprechen, ohne deren Mitwirkung auf die eine oder andere Weise DVI und DVILW nicht in ihrer jetzigen Form bestehen würden:

Tomas Rokicki für seine hervorragenden Vorarbeiten mit DVIPS. Donald E. Knuth für AFM2TFM und TEX. Friedrich Pukelsheim und Bernd Aulbach für unermüdliches Betatesten und Dutzende guter Ratschläge für die Anleitung, die wir leider nicht alle verwirklichen konnten. Stefan Lindner und Lutz Birkhahn für wertvolle Tips und Anregungen über mehrere Jahre hinweg.

Ferner danken wir allen anderen, die die Programme bisher benutzt haben und durch wertvolle Tips und hartnäckige Wünsche an Ingo Eichenseher mit zum jetzigen Funktionsumfang und Erscheinungsbild der Treiber beigetragen haben.

Sollten in diesem Text geschützte Markennamen vorkommen, möchten wir hier ausdrücklich darauf hinweisen, daß diese Namen durch den Eigentümer geschützt sind und lediglich als Referenzen auf die entsprechende Hard- und Software zu sehen sind (sozusagen kostenlose Reklame!). Als da wären: `UNIX`, `X11`, Motif, GEM, `TOS`, `MS-DOS`, `IBM`, Apollo, `SUN`, Sparc, `ST`, `PC`, AMIGA, Siemens, usw., usw.

Für Interessierte hier noch ein paar Worte zu den Autoren und ihren Funktionen. Ingo Eichenseher hat `DVI` und `DVILW` zunächst alleine für den Atari ST, später für `PC`s und Apollo Workstations entwickelt. Nachdem sowohl eine Anleitung als auch diverse Änderungen an der Treibern von Mitarbeitern und Studenten der Universität Augsburg lautstark gefordert wurden, kam Gerhard Wilhelms hinzu. Über einen Zeitraum von ca. 2 Jahren hat Ingo hauptsächlich die Treiber weiterentwickelt, während Gerhard kleine Änderungen und Erweiterungen einbaute, die Benutzerbetreuung und Erstellung des Handbuches übernahm. Als (bisher) letzter Mitarbeiter stieß Markus Zahn zum Team, um eine AMIGA-Version zu entwickeln. Inzwischen hat sich Markus sowohl durch Verbesserungen und Erweiterungen der Treiber, als auch bei Benutzerberatung und Erstellung des Handbuchs unentbehrlich gemacht (Sülz, sülz).

Für die vielen Tips und Anregungen von den Benutzern möchten wir an dieser Stelle nochmals unseren ausdrücklichen Dank aussprechen und gleichzeitig um Geduld bitten, falls sich die Bearbeitung aus unerfindlichen Gründen etwas verzögert oder der anonyme Fileserver (siehe Anhang C) mal wieder nicht erreichbar ist.

Kapitel 1. DVILW PostScript-Treiber

Der DVILW PostScript-Treiber wandelt TEX-DVI-Dateien in PostScript-Code um und schreibt diesen wahlweise in eine Datei bzw. schickt ihn auf PCs oder Atari STs über die serielle Schnittstelle oder ISDN-Schnittstelle an einen postscriptfähigen Laserdrucker. Der Programmaufruf ist wie folgt:

```
dvilw ⟨options⟩ ⟨dvifiles⟩
```

⟨dvifiles⟩ sind dabei Namen beliebig vieler DVI-Dateien bzw. ein gültiger Wild-Card wie *.dvi (bzw. #?.dvi auf dem Amiga). Auf diese Weise kann DVILW mehrere Dateien hintereinander bearbeiten. Die Angabe der Extension .dvi kann entfallen und wird in diesem Fall automatisch ergänzt. ⟨options⟩ ist eine Liste von Optionen, die auch leer sein kann. Zur Erklärung der möglichen Optionen lesen Sie bitte Abschnitt 1.2.

1.1 Konfiguration

Die wichtigsten Informationen, die DVILW braucht, werden über eine Konfigurationsdatei namens DVILW.OPT eingestellt. DVILW.OPT muß im Suchpfad (bzw. beim Amiga im Verzeichnis TeX:config oder in dem über die Environmentvariable TEXCONFIG angegebenen Ordner) stehen. Diese Konfigurationsdatei enthält jeweils in einer eigenen Zeile Zuweisungen an Variablen wie

```
pkpath=.:/usr/local/lib/tex/pk%h/%s.%dpk
```

Folgende Konfigurations-Variablen werden von DVILW benutzt:

PKPATH In dieser Variable wird der Name und der Pfad der benötigten Zeichensätze eingetragen. (Defaultauflösung für den Laserdrucker: 300 dpi.) Das Format der Zuweisung ist dabei durch die maximale Länge der Dateinamen für den verwendeten Computer bestimmt.

Der hier angegebene Format-String wird von DVILW ähnlich einem printf-Format-String übersetzt. Als Platzhalter können %d, %h, %m, %s und %v eingesetzt werden. Die Platzhalter haben folgende Entsprechungen:

%d	Zeichensatzgröße in dpi
%h	horizontale Geräteauflösung
%m	MAG___1.2 etc.
%s	Zeichensatzname, z.B. cmr10
%v	vertikale Geräteauflösung

Beispielsweise ergibt also eine Zuweisung wie

```
PKPATH=TeX:pk/%hx%v/%d/%s.%dpk
```

Zeichensatzpfade und -namen im Stil von

```
TeX:pk/300x300/360/cmr10.360pk,
```

die z.B. auf dem AMIGA im Rahmen der PasTEX-Implementation Sinn machen.

Auf Maschinen mit $8+3$-Dateinamen, wie z.B. Atari ST und PC, bei denen die einzelnen Auflösungsstufen der Zeichensätze jeweils in einem eigenen Ordner, der die Auflösung darstellt, mit jeweils gleichen Dateinamen abgespeichert werden, gibt es u.a. folgende Möglichkeiten der Einstellung:

```
pkpath=D:\FONTS\%m\%s.PK
pkpath=D:\FONTS\%d\%s.PK
```

Durch die jeweiligen Format-Definitionen werden dann die entsprechenden Pfade und Zeichensatznamen erzeugt:

```
D:\FONTS\MAG___1.2\CMR10.PK
D:\FONTS\360\CMR10.PK
```

Auf dem Atari ST ist dieses (neue) Zeichensatzcodierungsschema prinzipiell lauffähig, jedoch wegen Kompatibilität zur TEXshell von Heidrich, Maluschka und Kießling nicht in den ausführbaren Programmen enthalten. Wer gerne dieses Auswahlschema hätte, muß sich die Quellcodes besorgen und mit zusätzlich definierter Präprozessorvariable PK_FULLCONFIG neu übersetzen. Aber wer will schon auf die tolle TEXshell verzichten?

Das Zeichensatzauswahlschema auf dem Atari ist wie folgt. Es gibt zwei Möglichkeiten der Einstellung. Falls PKPATH *kein* %-Zeichen enthält, erzeugt DVILW Pfade wie

```
path\RES300.PS\MAG___1.2,
```

die zum „Lindner"-TEX auf dem Atari ST kompatibel sind. Ansonsten wird die komplette Zeichenkette als printf-Format-String übernommen. Die %-Anweisung muß dabei die Zeichensatzauflösung in dpi korrekt interpretieren. Durch

```
PKPATH=D:\FONTS\DVILW300\%d
```

werden dann z.B. Pfade wie

```
D:\FONTS\DVILW300\360
```

generiert. Falls mehrere Pfade durchsucht werden sollen, müssen die jeweiligen Pfadangaben durch ein ; getrennt werden.

VFPATH Diese Variable muß den Pfad auf die *.VF-Dateien enthalten, falls virtuelle Zeichensätze verwendet werden.

`TFMPATH` Diese Variable kann den Pfad auf die `*.TFM`-Dateien enthalten. Falls `DVILW` weder `PK`- noch `VF`-Dateien findet, werden für die nicht gefundenen Zeichensätze wenigstens leere Rechtecke der richtigen Größe freigelassen.

`IMGPATH` Diese Variable enthält den Pfad auf die `IMG`-Grafiken, die mit Hilfe des Befehls `\special{graphic img ... }` eingebunden werden.
→ 4.2.1

`DVIPATH` Diese Variable kann einen Pfad enthalten, der von `DVILW` zum Suchen nach `.dvi`-Dateien benutzt wird.

`GRPATH` Externe Dateien für den eingebauten Grafikbefehl `\special{gr input ... }` können in einem Unterverzeichnis abgelegt werden, das durch diese Variable spezifiziert wird.

`PSPATH` Diese Variable enthält den Pfad auf die PostScript-Dateien, die mittels `\special{ps ... }` eingebunden werden.
→ 4.2.1

`LINE` Diese Variable wählt für Geräte, die *nicht* unter `UNIX` arbeiten, die Ausgabeschnittstelle aus. Wenn die Variable nicht gesetzt ist, wird der PostScript-Code in eine Datei geschrieben, wobei der Name der `DVI`-Datei mit Endung `.ps` verwendet wird.

Falls die Variable gesetzt wird, muß auf dem `IBM PC` zuerst die Ausgabeschnittstelle angegeben werden. Optional können danach eine Telefonnummer eines `ISDN`-Modems und ein Faktor zur Prüfung auf Timeouts angegeben werden. Mit

```
LINE=2179;1
```

wird z.B. auf dem Atari ST über `ISDN`-Schnittstelle mit Zielmodem 2179 und Timeoutüberprüfung von einer Sekunde gedruckt.

```
LINE=COM2:
```

dagegen druckt auf dem `IBM PC` auf einen direkt an der zweiten seriellen Schnittstelle angeschlossenen Drucker.

SHOWFONTS Falls diese Variable einen Wert ungleich Null enthält, werden bei Erzeugung der `.ps`-Datei Informationen über die verwendeten Zeichensätze angezeigt.

TRACEMEM Falls diese Variable einen Wert ungleich Null enthält, werden bei Erzeugung der `.ps`-Datei Informationen über angeforderte und wieder freigegebene Speicherbereiche angezeigt.

MAXMEM Jeder Wert ungleich Null für diese Variable begrenzt den maximal angeforderten Speicher auf diesen Wert. Das Setzen dieser Variablen ist nur auf Mehrprozessrechnern sinnvoll, um unnötiges Pagen zu vermeiden.

MEMORY Falls diese Variable einen Wert ungleich Null enthält, wird die gesamte `.dvi`-Datei im Speicher gehalten, um ein schnelles Arbeiten zu ermöglichen.

HRESOLUTION Diese Variable dient zum Einstellen der horizontalen Auflösung des Ausgabegerätes. Der Wert muß in dpi angegeben werden. Die Voreinstellung ist 300.

VRESOLUTION Diese Variable dient zum Einstellen der vertikalen Auflösung des Ausgabegerätes. Der Wert muß in dpi angegeben werden. Die Voreinstellung ist 300.

HMARGIN Eingabe eines horizontalen Offsets in einer gültigen TEX-Längeneinheit.

VMARGIN Eingabe eines vertikalen Offsets in einer gültigen TEX-Längeneinheit.

MAGNIFICATION Eingabe eines Vergößerungsfaktors, wobei ein Wert von 1000 der Originalgröße entspricht, ein Wert von 1200 um den Faktor 1.2 vergrößert, etc. Die Vergrößerung wird durch Verwendung entsprechend skalierter Zeichensätze erreicht, deshalb sind als Werte Potenzen von 1.2 vorzuziehen!

DENSITY Angabe des Dunkelfaktors für `IMG`-Bilder.

COPIES Anzahl der Ausdrucke pro Seite.

LANDSCAPE Setzen dieser Variable führt zum Drehen des Ausdrucks um 90°. Evtl. muß die Seite noch durch `HMARGIN` und `VMARGIN` Anweisungen richtig positioniert werden.

CALLMF Diese Variable existiert nur auf Amiga-Implementationen und enthält den Namen eines `ARexx`-Scripts, das einen nicht vorhandenen Zeichensatz evtl. zur Laufzeit generiert. Bisher steht nur das `ARexx`-Script `MakePkFont.rexx` zur Verfügung, das im `REXX:`-Verzeichnis stehen sollte.

1.2 Optionen

Viele Konfigurations-Einstellungen und einige nicht-permanente Einstellungen können über Kommandozeilenoptionen eingestellt werden, wobei die Eingabe über die Kommandozeile Vorrang vor den Einstellungen der Konfiguration haben. Optionen beginnen stets mit [-], gefolgt von der Option selbst und möglichen Parametern. Zwischen [-] und der Option darf kein Leerzeichen stehen, dagegen sind zwischen Option und Parameter Leerzeichen erlaubt. Folgende Optionen stehen zur Verfügung:

-*o*⟨*list*⟩ Falls nicht das gesamte Dokument gedruckt werden soll, sondern nur einzelne Seiten, können diese mit der -*o*-Option ausgewählt werden. ⟨*list*⟩ ist eine Seitenliste, die beispielsweise folgende Form hat:

```
4,9-12,18,99-103
```

Mit dieser Liste werden nur die Seiten 4,9,10,11,12,18,99,100,101,102 und 103 des Dokumentes gedruckt. Durch Angabe von *-o start:stop*, *-o range:step* oder *-o start:stop:step* kann alternativ ein Seitenbereich bzw. ein Seitenbereich mit Schrittweite angegeben werden.

-*b*⟨*n*⟩ Einstellen der ersten Seite ⟨*n*⟩, die gedruckt werden soll. Im Prinzip ist diese Option redundant, da in -*o* enthalten, aber wegen Kompatibilität zur TEXshell auf dem Atari ST trotzdem sinnvoll. Gleiches gilt für:

-*e*⟨*n*⟩ Einstellen der letzten Seite ⟨*n*⟩, die gedruckt werden soll.

-*c*⟨*n*⟩ Es werden ⟨*n*⟩ Kopien gedruckt.

-n Dieser Schalter verhindert das direkte Drucken, auch wenn die Konfigurations-Variable `LINE` gesetzt ist. Es wird grundsätzlich eine Datei erzeugt.

-y Dieser Schalter bewirkt, daß jede Ausgabezeile des PostScript-Codes mit einer Carriage Return + Line Feed Kombination abgeschlossen wird. Normalerweise genügt für PostScript ein `LF` als Zeilenende. Solche Dateien können allerdings von einigen dummen `PC`-Editoren wie dem Norton Editor nicht gelesen werden.

-d⟨line⟩ Entspricht der `LINE`-Variablen.

-m⟨mag⟩ Setzt die Vergrößerung auf ⟨*mag*⟩, wobei 1000 der Standardgröße entspricht, 1200 einer Vergrößerung um den Faktor 1.2, usw.

-f⟨path⟩ Entspricht der `PKPATH`-Variablen.

-g⟨path⟩ Entspricht der `IMGPATH`-Variablen.

-q Diese Option bewirkt eine Bildschirmmeldung über jeden beim Ausdruck verwendeten Zeichensatz. Entspricht der `SHOWFONTS`-Variablen.

-r Dieser Schalter bewirkt eine Ausgabe der Seiten in umgekehrter Reihenfolge.

-u Ausgabe der Seiten in physikalischer Reihenfolge (Voreinstellung).

-a⟨name⟩, -p⟨name⟩ Die Ausgabedatei wird in ⟨*name*⟩ umbenannt. Wird diese Option nicht verwendet, erhält die erzeugte PostScript-Datei denselben Namen, wie die erste `DVI`-Datei in der ⟨dvifiles⟩-Liste, nur mit der Extension `.ps`.

-h⟨dimen⟩ Einrückung der Druckseite nach rechts oder links (negative Werte). ⟨*dimen*⟩ ist eine gültige Längenangabe, wie sie auch von TEX verstanden wird, wie z.B. $1pt$, $2.7cm$, usw. Der Parameter arbeitet additiv relativ zu *hmargin*.

-v⟨dimen⟩ Einrückung der Druckseite nach unten oder oben (negative Werte). ⟨*dimen*⟩ ist eine gültige Längenangabe, wie sie auch von TEX verstanden wird, wie z.B. $1pt$, $2.7cm$, usw. Der Parameter arbeitet additiv relativ zu *vmargin*.

-w Alle Rückmeldungen des Laserdruckers auf dem Bildschirm anzeigen.

-s⟨n⟩ Alle ⟨n⟩ Sekunden wird eine Statusmeldung des Druckers angezeigt. ⟨n⟩ ist eine positive Fließkommazahl.

-zr⟨integer⟩ Mit diesem Befehl kann die Druckerauflösung in `dpi` angegeben werden. Voreingestellt ist der Wert 300.

-zt⟨path⟩ Dieser Befehl erlaubt die Angabe des Suchpfades für `TFM`-Dateien, die allerdings nur dann benötigt werden, wenn PostScript-Zeichensätze verwendet wurden oder `PK`-Zeichensatzdateien fehlen. Der Pfad kann auch über die Konfigurations-Variable `TFMPATH` angegeben werden.

-zi⟨ext⟩ Dieser Befehl dient zur Eingabe der Extension des Zeichensatzgeräteordners auf dem Atari ST. ⟨*ext*⟩ besteht aus bis zu drei Buchstaben.

-j Diese Option unterdrückt die Ausgabe einer Meldung im Fehlerfall und stellt das Warten auf Eingabe einer Taste ab. Sehr sinnvoll in Verbindung mit

einer Shell, die diese Fehler von sich aus bearbeitet (z.B. die TEXshell von Heidrich/Maluschka/Kießling auf dem Atari ST).

-t Falls dieser Schalter gesetzt ist, erfolgt die Ausgabe auf `stdout`. Diese Option dient hauptsächlich zum Drucken auf `UNIX`-Maschinen, wo diese Ausgabe mittels einer Pipe an das entsprechende Druckprogramm weitergeleitet wird.

-k ⟨filename⟩ Dieser Schalter dient zur Auswahl einer alternativen Prologdatei. dies kann z.B. zum Ausdruck im Landscape-Modus notwendig sein.

-l Dieser Schalter führt zu einer um 90° gedrehten Ausgabe.

1.3 Arbeitsweise von DVILW

`DVILW` überprüft, ob die `DVI`-Datei(en) vorhanden ist/sind, versucht, falls gewünscht, die Verbindung zum Drucker aufzubauen und lädt dann soviele Zeichensätze als möglich. Falls Zeichensätze fehlen, wird eine zur TEXshell von Heidrich-Maluschka-Kießling kompatible Datei `missing.fnt` erstellt, mit der die fehlenden Zeichensätze automatisch erstellt werden können. Auf `UNIX`-Geräten und Amigas ist diese Datei ein Shell-Script. Durch Eingabe von

```
./missing.fnt postscript
```

auf `UNIX`-Geräten, bzw. durch das Kommando

```
execute missing.fnt CanonCX
```

auf Amigas, kann dieses gestartet werden. Auf PC-kompatiblen Rechnern wird ein Batchjob mit Namen `MISSING.BAT` erstellt, der durch Aufruf weiterer Batchjobs die Zeichensätze automatisch erstellt. Bei fehlenden Zeichensätzen wird auf `UNIX`-Rechnern die Bearbeitung abgebrochen, ansonsten kann man mit leeren Bitmustern die Bearbeitung fortsetzen.

Auf dem Amiga kann alternativ ein in der Konfigurationsvariablen `CALLMF` angegebenes externes `ARexx`-Programm gestartet werden , das dann entweder den Zeichensatz generiert, oder eine entsprechende Batchdatei erzeugt/erweitert. Im Fall von `MakePKFont.rexx` wird versucht, den fehlenden Font direkt zu erzeugen.

Die DVI-Datei wird in PostScript-Code umgewandelt und an den Drucker geschickt bzw. in die Ausgabedatei geschrieben. Während der Umwandlung wird die gerade bearbeitete Seite durch `[x` angezeigt, wobei `x` natürlich der Wert einer der TEX-Zähler `\count0` bis `\count9` ist. Ist die Seite fertig bearbeitet, wird die schließende Klammer `]` gedruckt. Beachten Sie bitte, daß die angezeigten Nummern `x` durchgehend nach der physikalischen Reihenfolge der Seiten in der `DVI`-Datei angezeigt werden und der tatsächlichen Seitenzahl im Dokument nicht unbedingt entsprechen!

Falls Sie aus irgendeinem Grund die Umwandlung abbrechen wollen, drücken Sie auf `PCs` und dem Atari ST Esc, bzw. auf `UNIX`-Systemen und dem Amiga Ctrl + C. `DVILW` bricht daraufhin sofort die Arbeit ab!

Sollten Sie auf UNIX-Geräten mit `missing.fnt` Zeichensätze erzeugt haben, stehen diese im aktuellen Verzeichnis. Um diese Zeichensätze bei folgenden Arbeiten mit TEX zur Verfügung zu haben, müssen Sie die Dateien, die auf `pk` enden, in den Ordner

```
/usr/local/lib/tex/pk300
```

kopieren (bzw. wo auch immer in Ihrer TEX-Installation die Zeichensätze für PostScript installiert sind) und die `pk`-Dateien im aktuellen Ordner löschen.

Auf Amiga-Computern werden die erzeugten Fonts direkt im ersten in der Konfigurationsvariablen `PKPATH` angegebenen Verzeichnis abgelegt. Die durch METAFONT erzeugten temporären Dateien werden hier automatisch gelöscht.

Für alle Systeme gilt, daß Sie die temporären Dateien, die bei der Zeichensatzerzeugung angelegt wurden, am Besten löschen, falls dies nicht schon, wie auf dem Amiga, geschehen ist. Es handelt sich um Dateien, die auf `gf` enden, sowie um Dateien mit Endungen `.log` und `.tfm`.

1.4 Installation

Die Installation von `DVILW` gestaltet sich trotz der vielfältigen Einstellmöglichkeiten sehr einfach. Die ausführbaren Programme `dvilw` (`UNIX` und Amiga), `dvilw.ttp` (Atari ST) bzw. `dvilw.exe` (`MS-DOS`) werden in irgend einen Ordner kopiert, der im Systemsuchpfad enthalten ist. Unter `UNIX`, `MS-DOS` und auf dem Atari ST kopieren Sie dorthin ebenfalls die Dateien `dvilw.ps` und `dvilw.map`, beim Amiga kommen diese beiden Dateien in den Ordner `TeX:config` oder in das in der Environmentvariablen `TEXCONFIG` angegebene Verzeichnis. Will man auf dem Amiga nichtvorhandene Zeichensatzgrößen automatisch erzeugen lassen, so ist zusätzlich das `ARexx`-Script `MakePKFont.rexx` in das `Rexx:` Verzeichnis zu kopieren.

Für den Amiga wird es in der näheren Zukunft ein Installationsscript für den `Installer` von Commodore geben, mit dem dann die oben beschriebenen Aktionen nicht mehr zu Fuß erledigt werden müssen.

Zum Arbeiten mit reinen Texten ohne Grafik muß dann nur noch die Konfigurationsdatei `dvilw.opt` mit mindestens einem Eintrag, nämlich der Konfigurationsvariablen `PKFONTS` entsprechend der Organisation der Zeichensatzdateien auf der Festplatte, angelegt werden. Bei `UNIX` ist der übliche Pfad

```
./%s.%dpk:/usr/local/lib/tex/pk%h/%s.%dpk
```

voreingestellt, wobei alle Auflösungsstufen eines Zeichensatzes in einem dieser beiden Ordner liegen. Bei Systemen mit 8 Zeichen langen Dateinamen und 3 Zeichen langen Erweiterungen sind die Auflösungsstufen der Zeichensätze naturgemäß nicht im Dateinamen unterzubringen. Man hilft sich

dahingehend, daß die Zeichensatznamen nur aus dem Basisnamen bestehen — z.B. `cmr10` — und die Endung `.pk` erhalten. Die Auflösungsstufen der Zeichensätze werden auf eine Ordnerhierarchie abgebildet, wobei die Ordnernamen die Vergrößerungsstufe — z.B. `mag____1.728` — oder die Auflösung in dpi — z.B. `518` — bezeichnen. Diese Auflösungsstufe errechnet sich zu Gerätegrundauflösung * Vergrößerungsstufe, hier also 300 dpi * `1.728`.

Auf Rechnern ohne `UNIX`- oder AMIGA-Betriebssystem steuert dann die Variable `LINE` die Ausgabe auf Drucker, Modem oder Datei.

Sollen schließlich noch Fremdgrafiken eingebunden werden, die sich nicht im aktuellen Arbeitsordner befinden, sind die Variablen `PSPATH` und `IMGPATH` entsprechend zu setzen.

Alle übrigen Variablen können selbstverständlich nach Wunsch umgesetzt werden, falls sie Ihr TEX-System anders konfiguriert haben.

1.5 `DVIPS` oder `DVILW`?

Abschließend noch ein paar Worte zum POSTSCRIPT-Treiber `DVIPS` von Tomas Rokicki. Beim aufmerksamen Durchlesen der Anleitung wird dem Leser sicher aufgefallen sein, daß bei beiden Treiber viele Leistungsmerkmale identisch sind und es stellt sich die Frage, welcher der beiden Treiber für welchen Personenkreis sinnvoll ist.

Zunächst zu `DVIPS`. Dieser Treiber zeichnet sich durch ökonomische Speicherausnutzung sowohl im Computer als auch im Laserdrucker aus. Wer also in einem der Geräte mit Speicher mager ausgestattet ist, für den dürfte `DVIPS` die bessere Wahl sein.

`DVILW` dagegen benötigt — bedingt durch die zusätzlich vorhandenen Grafikmöglichkeiten — mehr Hauptspeicher und zudem durch Verwendung eines anderen POSTSCRIPT-Prolog wesentlich mehr Speicher im Drucker.

Dagegen stehen die Vorteile der Grafikmöglichkeiten, vor allem im Zusammenhang mit identischem Bildschirmpreview über `DVI` und die deutlich höhere Geschwindigkeit des Ausdrucks.

Wer also 2 MB Speicher im POSTSCRIPT-Drucker installiert hat, sollte ohne Zögern zu `DVILW` greifen, auch wenn er keine Grafik benötigt. Was den Speicherbedarf im Computer angeht, sind ab Version 2.53 umfangreiche Maßnahmen getroffen worden, um auch mit wenig Speicher einen Ausdruck erzeugen zu können.

Kapitel 2. DVI-Bildschirmtreiber

Der Bildschirmtreiber DVI ist momentan auf dem Atari ST für Standard-GEM-Fenster (auch Großbildschirme), auf Workstations unter X11/Motif, auf Apollo Workstations unter GPR, auf dem Commodore AMIGA ab AMIGA OS 2.04, sowie auf dem IBM PC mit gängiger Grafikkarte implementiert. Für alle Systeme werden Zeichensätze der Auflösung 101×101 dpi empfohlen, jedoch sind natürlich auch andere Auflösungen problemlos möglich.

2.1 DVI GEM-Version, Atari

Die GEM-Version von DVI ist fast durchgängig per Maus bequem zu bedienen. Die gewünschten Optionen werden mit den GEM-typischen Drop-Down-Menus mit Hilfe der Maus eingestellt und können auch in einer Parameterdatei gespeichert werden. Der Aufruf von DVI.PRG ist wie folgt:

```
dvi [-o<optionfile>] [dvifile] [optionfile]
```

Alle Parameter sind optional, da auch interaktiv über Menüs einstellbar. Falls jedoch Parameter angegeben werden, ist die Reihenfolge zwingend vorgeschrieben. Wenn erwünscht, müssen Optionen — durch ein [-]-Zeichen eingeleitet — zuerst angegeben werden. Danach können ein oder zwei Dateinamen folgen, wobei der erste immer als Name der zu bearbeitenden .DVI-Datei angesehen wird. Ein zweiter Dateiname wird als Name einer Optionsdatei interpretiert. Bei Dateinamen werden fehlende Extensionen .DVI bzw. .DVO automatisch ergänzt. Als Option steht momentan nur *-o* zur Verfügung. Danach muß ein Dateiname einer Optionsdatei angegeben werden, der nicht durch ein Leerzeichen von der Option getrennt sein darf. Alle anderen Optionen werden ignoriert!

Falls eine Optionsdatei sowohl über *-o* als auch dem .DVI-Dateinamen folgend angegeben wird, hat letztere Datei den Vorrang. Die Einstellungen der mit *-o* angegebenen Datei werden nicht übernommen.

Der Arbeitsbildschirm von DVI ist in Bild 2.1 zu sehen.

Der Bildschirm enthält die Menüleiste zum Einstellen der Optionen und ein Fenster zum Anzeigen von Meldungen. Die einzelnen Einträge der Menüleiste werden nachfolgend erklärt, auch wenn eigentlich alle Einträge noch bei der Kommandozeilenversion in Abschnitt 2.4 ausführlich erklärt werden.

Abbildung 2.1: Der Arbeitsbildschirm von DVI

2.1.1 Der Desk Titel

About DVI ...
accessory 1
accessory 2
accessory 3
accessory 4
accessory 5
accessory 6

About DVI ... Druckt das obligatorische Formular mit Informationen über DVI.

Accessories GEM unterstützt bis zu 6 im Hintergrund wartende Programme, die sich über die Accessory-Einträge aktivieren lassen. Siehe auch Anhang B.

2.1.2 Der File Titel

Choose DVI	[C]
Load Options	[L]
Save Options	[S]
Logfile	
Quit	[Q]

Choose DVI Durch Anklicken dieses Menüpunktes oder Drücken der C-Taste wird eine Dateiauswahlbox zum Wählen der DVI-Datei angezeigt. Alternativ kann der Dateiname auch bei Aufruf in der Kommandozeile übergeben werden.

Load Options Dieser Menütitel bzw. die L-Taste bringen eine Dateiauswahlbox zum Wählen einer Optionsdatei zur Anzeige. Optionsdateien von DVI erkennt man an der *.DVO Endung. Die Standardoptionen sollten in der Datei DVI.DVO gespeichert sein. Diese Optionsdatei wird beim Starten von DVI automatisch geladen, wenn keine Optionsdatei über die Kommandozeile übergeben wurde und sollte zweckmäßigerweise die Bildschirmoptionen enthalten. Bei

einigen speziellen Namen von Optionsdateien kann man sich das Anwählen über Menü ersparen. Die Optionsdateien `F1.DVO`–`F10.DVO` können direkt durch Drücken der Tasten F1–F10 geladen werden.

Save Options Durch diesen Eintrag bzw. durch Drücken der S-Taste wird eine Dateiauswahlbox zum Wählen des Dateinamens der zu speichernden Optionsdatei angezeigt. Alle aktuellen Optionseinstellungen werden in diese Optionsdatei gespeichert.

Logfile Alle Meldungen, die im Meldungsfenster erscheinen, können mittels dieser Option in eine Datei geschrieben werden, die mittels einer Dateiauswahlbox ausgewählt wird.

Quit Anklicken dieses Eintrags bzw. Drücken von Q beendet das Programm.

2.1.3 Der Print Titel

Next Page	[N]
Previous Page	[P]
Format from	[F]
Cycle Windows	[W]
Lower Limit	^L
Upper Limit	^U
Magnification	^M
Copies	^C

Next Page Anklicken dieses Eintrags bzw. Drücken von N startet den Aufbau der ersten bzw. nächsten Bildschirmseite in einem eigenen Fenster. Nach evtl. längerer Wartezeit — bedingt durch Grafikoperationen, die sehr rechenintensiv sind — erscheint die Seite im Grafikfenster. Gleichzeitig wird ein Meldungsfenster geöffnet, in dem entsprechend den eingestellten Optionen diverse Operationen wie bearbeitete Seite, Laden und Auslagern von Zeichensätzen, Anfordern und Freigeben von Speicher, etc. protokolliert werden. Beim Aufbau der Grafikseite wird dieses Fenster automatisch sichtbar gemacht und nach Aufbau der Grafikseite von dieser verdeckt. Durch Ändern der Fenstergrößen kann man selbstverständlich beide Seiten gleichzeitig darstellen. Falls Ihnen der ständige Fensterwechsel nicht zusagt, können Sie dies im Menütitel **Setup** mit der Funktion **Top Windows** ausstellen.

Previous Page Anklicken dieses Eintrags bzw. Drücken von P startet den Aufbau der vorherigen bzw. letzten Bildschirmseite.

Format from Dieser Eintrag bzw. Drücken der Taste F ermöglicht die Auswahl einzelner Seiten des Dokuments und startet nach der Eingabe den Ausdruck. Die Seiten werden mit bis zu drei Parametern ausgewählt. Die erste Zahl gibt die Startseite an, die zweite die Stopseite und die dritte die Schrittweite zwischen den Seiten. Mit den Parametern `4, 8, 2` werden nur die Seiten 4, 6 und 8 gedruckt.

Cycle Windows Durch diese Funktion kann man wechselweise das Meldungsfenster und das

Grafikfenster sichtbar machen, falls sie sich gegenseitig überdecken.

Lower Limit Kontrolle der Ausdruckseiten über die TEX-internen Seitenzähler \count0 bis \count9.

Upper Limit Kontrolle der Ausdruckseiten über die TEX-internen Seitenzähler \count0 bis \count9.

Magnification Dieser Eintrag bzw. gleichzeitiges Drücken der Tasten CONTROL und M dient zum Einstellen des Vergrößerungsfaktors. Ein Wert von 1000 entspricht Originalgröße, ein Wert von 1200 vergrößert um den Faktor 1.2, usw. Beachten Sie aber bitte, daß die Vergrößerung durch Verwendung größerer Zeichensätze bewirkt wird, die natürlich vorhanden sein müssen. Deshalb sind wohl nur die üblichen Vergrößerungsstufen in Schritten von 1.2^i, $i \in \{ 0, 1/2, 1, \dots \}$ praktikabel.

Copies Dieser Eintrag bzw. gleichzeitiges Drücken der Tasten CONTROL und C dient zum Einstellen der Anzahl Kopien pro Seite. Beachten Sie, daß diese Option wesentlich schneller arbeitet, als mehrmaliges Ausdrucken, da für die Kopien die Interpretation der DVI-Datei entfällt!

2.1.4 Der Margins Titel

H-Offset	
V-Offset	
H-Spread	
V-Spread	
H-Margin	
V-Margin	
Width	^W
Height	^H

H-Offset Seitenverbreiterung links.

V-Offset Seitenverlängerung oben.

H-Spread Seitenverbreiterung rechts.

V-Spread Seitenverlängerung unten.

H-Margin Einrückung der Druckseite nach rechts.

V-Margin Einrückung der Druckseite nach unten.

Width Seitenbreite. Falls mit Null belegt, wird der Eintrag der DVI-Datei übernommen.

Height Seitenhöhe. Falls mit Null belegt, wird der Eintrag der DVI-Datei übernommen.

2.1.5 Der Devices Titel

Screen	`1S`
File	`1F`
NEC P6 Low	`1L`
NEC P6 Mid	`1M`
NEC P6 High	`1H`
FX 80	`1X`
HP Laserjet	`1J`
HP Laserjet Lo	`1K`
Canon BJ300	`1B`
Print on File	`1P`
Null Device	`1N`

Screen Ausgabe auf Bildschirm in variabler Auflösung. Auch über die Tastenkombination [ALTERNATE] + [S] einstellbar.

File Ausgabe auf Datei zur Weiterverarbeitung mit `DISPLAY` (siehe auch Anhang B).

NEC P6 Low Ausgabe auf 24-Nadler mit NEC-kompatiblen Befehlssatz in 180×180 dpi.

NEC P6 Mid Ausgabe auf 24-Nadler mit NEC-kompatiblen Befehlssatz in 360×180 dpi.

NEC P6 High Ausgabe auf 24-Nadler mit NEC-kompatiblen Befehlssatz in 360×360 dpi.

FX 80 Ausgabe auf 9-Nadler mit Epson-kompatiblen Befehlssatz in 240×216 dpi.

HP Laserjet Ausgabe auf Laserdrucker mit HP Laserjet II-kompatiblen Befehlssatz in 300×300 dpi.

HP Laserjet Lo Ausgabe auf Laserdrucker mit HP Laserjet II-kompatiblen Befehlssatz in 100×100 dpi zur Previewzwecken.

Canon BJ300 Ausgabe auf entsprechenden Laserdrucker.

Print on File Die Ausgabe erfolgt nicht auf die Schnittstelle zum Drucker, sondern auf eine Datei. Diese kann dann auf einem Rechner ausgedruckt werden, der über den eingestellten Drucker verfügt, aber kein TEX-System installiert hat.

Null Device Bei diesem Device erfolgt keine Ausgabe. Man kann dadurch schnell testen, ob auch alle Zeichensätze vorhanden sind, oder ob in der `.dvi`-Datei ein Fehler ist.

2.1.6 Der Setup Titel

Top Windows	
Maximum Memory	
Bitmap Memory	
Clippath Memory	
DVI in Memory	
Trace Fonts	
Trace Memory	
Trace Char's	
PK-Font	Path
VF-Font	Path
TFM-File	Path
DVI-File	Path
IMG-Image	Path
GR-Include	Path

Top Windows Dieser Schalter beeinflußt die „Fensterpolitik“. Man kann den ständigen Wechsel zwischen Meldungs- und Grafikfenster an- und wieder ausschalten.

Maximum Memory Einstellung der maximal angeforderten Speichermenge insgesamt. Sinnvoll bei Verwendung von Multitasking-Systemerweiterungen wie Mint, um noch Arbeitsspeicher für andere Prozesse freizuhalten. Ein Wert von Null (Default) bewirkt das Ausschöpfen der Speicherkapazität.

Bitmap Memory Beschränkung des maximal angeforderten Speicherbereichs für die Grafikseite.

Clippath Memory Beschränkung des maximal angeforderten Speicherbereichs für den Clipalgorithmus der eingebauten Grafik.

DVI in Memory Setzen dieses Schalters bewirkt das Laden der gesamten `.dvi`-Datei in den Arbeitsspeicher, um die Übersetzung zu beschleunigen.

Trace Fonts Durch diesen Schalter wird das Protokollieren eingeladener und ausgelagerter Zeichensätze ein- und ausgeschaltet.

Trace Memory Durch diesen Schalter wird das Protokoll über angeforderten und freigegebenen Speicher an- und ausgeschaltet.

Trace Char's Durch diesen Schalter wird das Protokoll über *jedes* gesetzte Zeichen an- und ausgeschaltet.

PK-Font Path Einstellung der Fontpfade und -namen. Wie schon einmal bei `DVILW` in Abschnitt 1 erläutert, kann im Zusammenhang mit den Platzhaltern `%d, %h, %m, %s` und `%v` der Name und Pfad der verwendeten Zeichensätze dargestellt werden. Die Platzhalter haben folgende Entsprechungen:

%d	Zeichensatzgröße in dpi
%h	horizontale Geräteauflösung
%m	`MAG____1.2` etc.
%s	Zeichensatzname, z.B. `cmr10`
%v	vertikale Geräteauflösung

Beispielsweise ergibt also eine Einstellung wie

```
FONTS/LASER/%d/%s.pk
```

Zeichensatzpfade und -namen im Stil von

```
FONTS/LASER/111/cmr10.pk.
```

Auf dem Atari ST ist dieses (neue) Schema aus Gründen der Kompatibilität zur TEXshell von Heidrich, Maluschka und Kießling nicht in den ausführbaren Programmen enthalten, sondern man muß — wenn gewünscht — die Sourcen mit definierter Präprozessorvariable `PK_FULLCONFIG` neu übersetzen. Das jetzige Schema ist wie folgt. Enthält die Zeichenkette *kein* %-Zeichen, führt dies zur Generierung von Pfaden wie `MAG___1.440` und ist kompatibel zum „Lindner"-TEX, ansonsten wird diese Zeichenkette als `printf`-Formatstring interpretiert und muß die Vergrößerungsstufen der Zeichensätze korrekt auf die tatsächliche Ordnerstruktur abbilden. Durch Angabe von

```
FONTS/LASER/%d
```

werden Pfade wie `FONTS/LASER/360` gebildet.

`VF`-Font Path Einstellung des Pfades auf die virtuellen Zeichensatzinformationen.

`TFM`-File Path Einstellung des Pfades auf die Zeichensatzmetrikdateien, die von `DVI` herangezogen werden, falls die `PK`-Dateien fehlen. Auf diese Weise können wenigstens leere Boxen in der richtigen Breite gesetzt werden.

`DVI`-File Path Einstellung des Pfades auf die `DVI`-Dateien.

`IMG`-Image Path Einstellung des Pfades auf die `IMG`-Grafiken.

GR-Include Path Einstellung des Pfades auf Dateien, die Grafikbefehle des Treibers enthalten und über `\special{gr input ... }` nachgeladen werden.

2.1.7 Der Options Titel

H-Resolution V-Resolution
Landscape Separate Thin out Pictures Eject Single Sheet
Density

H-Resolution Geräteauflösung horizontal in dpi.

V-Resolution Geräteauflösung vertikal in dpi.

Landscape Drehung des Ausdrucks um 90°. Natürlich darf die Seite nicht zu lang sein

und für die Outputroutinen fx80 und p6mid sind spezielle Zeichensätze nötig (216×240 dpi und 180×360 dpi, logisch!).

Separate Falls dieser Schalter gesetzt ist, werden bei Nadeldruckern zusätzliche Druckdurchgänge gestartet, um Ausdruck von nebeneinanderliegenden Nadeln zu trennen. Verdoppelt Druckzeit, aber erhöht Qualität!

Thin out Durch diese Option wird die erzeugte Seitenansicht „ausgedünnt“. Diese Option ist ein Ersatz für „Separate“, führt jedoch nicht zu zwei Druckdurchgängen. Die Qualität ist jedoch schlechter.

Pictures Falls dieser Schalter gesetzt ist, werden die `IMG`-Grafiken mit angezeigt, ansonsten erscheint nur ein leerer Rahmen. Ohne Bilder werden die Druckseiten deutlich schneller aufgebaut!

Eject Seitenvorschub nach jeder Druckseite bei gesetztem Schalter.

Single Sheet Falls Einzelblattverarbeitung mit manuellen Einspannen des Papiers beabsichtigt ist, wird durch diese Option der Druckvorgang am Seitenende jeweils unterbrochen und kann dann nach Einspannen des neuen Blatts fortgesetzt werden.

Density Faktor zur Einstellung der Druckschwärze der `IMG`-Grafiken. Der Wert sollte zwischen 0.0 und 1.0 liegen, wobei 0.0 Bilder weiß auf weiß erzeugt und 1.0 sehr dunkle Bilder. Empfehlenswert ist 0.5 auf Laserdruckern, 0.02 auf Nadeldruckern.

2.2 DVI-X11/Motif-Version

Die X11/Motif-Version von `DVI` hält sich in Aussehen und Bedienung eng an den durch die GEM-Version vorgegebenen Standard. Der Aufruf ist wie folgt:

```
dvi [-o<optionfile>] [dvifile]
```

Alle Parameter sind optional, da auch interaktiv über Menüs einstellbar. Falls jedoch Parameter angegeben werden, ist die Reihenfolge zwingend vorgeschrieben. Wenn erwünscht, müssen Optionen — durch ein `-`-Zeichen eingeleitet — zuerst angegeben werden. Danach kann der Dateiname der `DVI`-Datei folgen. Beim Dateinamen werden fehlende Extensionen `.DVI` bzw. `.DVO` automatisch ergänzt. Als Option steht momentan nur *-o* zur Verfügung. Danach muß ein Dateiname einer Optionsdatei angegeben werden, der nicht durch ein Leerzeichen von der Option getrennt sein darf. Alle anderen Optionen werden ignoriert! Falls keine Optionsdatei beim Aufruf angegeben wird, wird automatisch die Standardoptionsdatei `DVI.DVO` geladen, sofern im Suchpfad vorhanden.

Der Arbeitsbildschirm von `DVI` in einer typischen `X11`-Umgebung ist in Abbildung 2.2 zu sehen.

Nach dem Starten von `DVI` werden zwei Fenster geöffnet, das Meldungsfenster mit Menüleiste und das Grafikfenster für die Darstellung der TEX-Seite.

Abbildung 2.2: Typische X11-Oberfläche mit `DVI`

Zunächst sollte das Grafikfenster mit der Maus in die gewünschte Position und auf die richtige Größe gebracht werden. Falls ein Dateiname auf der Kommandozeile übergeben wurde, wird schon die erste Seite des Dokumentes angezeigt, sofern die Eintragungen der Default-Konfigurationsdatei stimmen. Falls keine Anzeige erscheint, sollten die Einstellungen der `Setup`- und `Options`-Menüs berichtigt werden und eine neue `dvi.dvo`-Konfigurationsdatei erzeugt werden.

Im Grafikfenster kann man durch Drücken *und Halten* der linken Maustaste den Bildausschnitt verschieben (deshalb auch keine Scrollbalken). Drücken der rechten Maustaste bringt folgendes Popup-Menü:

Next Page
Prev Page
Goto Page ...
Update Page
Quit

Die möglichen Aktionen sind bis auf „Update Page“ wohl selbsterklärend. Ein Update der Seite bedeutet, daß sämtliche Änderungen, die zwischenzeitlich gemacht wurden, berücksichtigt werden, d.h. neue Offsets oder z.B. eine im Multitasking neu erzeugte `DVI`-Datei werden beachtet, und die Seite wird neu aufgebaut!

Die Menüleiste ist der Leiste der GEM-Version sehr ähnlich, als einziger Unterschied sind in der X11/Motif-Version einige Einträge der Menütitel nicht

vorhanden, weil unnötig bzw. weil durch eigene grafische Bedienelemente repräsentiert[1]. Die einzelnen Titel der Menüleiste werden im folgenden nur kurz vorgestellt, für die Erklärung der Unterpunkte lesen Sie bitte in Abschnitt 2.1 unter den analogen Überschriften nach. Gleiches gilt für die grafischen Bedienelemente, die jeweils ihre Entsprechung in Menüpunkten der GEM-Version haben. Durch andauernde Entwicklung ändert sich das Aussehen der Oberfläche ständig, so daß leichte Abweichungen zum Screendump möglich sind, die aber nichts an der Funktionalität ändern!

2.2.1 Der File Titel

Select File
Load Options Save Options
Logfile
Quit

2.2.2 Der Print Titel

Format Pages Magnification

2.2.3 Der Setup Titel

Clippath Memory Maximum Memory
PK-Font Path VF-Font Path TFM-File Path IMG-Image Path GR-Include Path DVI-File Path

2.2.4 Der Margins Titel

H-Offset V-Offset
H-Spread V-Spread

2.2.5 Der Options Titel

H-Resolution V-Resolution

[1] Bedingt durch die deutlich höhere Bildschirmauflösung von X11/Motif-Workstations kann natürlich wesentlich mehr Platz für die Benutzerführung verwendet werden

2.3 DVI AMIGA-Version

Die AMIGA-Version ist eine im gewohnten AMIGA-Komfort zu bedienende Version von `DVI`. Sie hält sich nicht ganz an die von der GEM- und X11/Motif-Version vorgegebene Benutzerführung, ist aber nicht weniger intuitiv zu bedienen. So sind z.B. einige Menüs etwas umgruppiert, verschwunden, hinzugekommen oder zusammengefaßt. Zu beachten ist, daß der AMIGA-Bildschirmtreiber im Gegensatz zu `dvilw` AMIGA OS 2.04 oder größer zwingend vorraussetzt.

Der Arbeitsbildschirm von `DVI` in einer typischen `Intuition`-Umgebung ist in Abbildung 2.3 zu sehen.

Abbildung 2.3: Typische Intuition-Oberfläche mit `DVI`

Der `DVI`-Treiber für den AMIGA unterstützt im Gegensatz zu z.B. der GEM-Version keine verschiedenen Optionsdateien. Die Konfiguration kann auf verschiedene Arten erfolgen: `DVI` ermittelt seine Startkonfiguration aus der Optionsdatei `dvi.opt`, die sich wahlweise im Verzeichnis `TeX:config` oder in dem durch die Environmentvariablen `TEXCONFIG` spezifizierten Ordner befinden muß. Die hierdurch festgelegten Eigenschaften können in der `Shell` durch Kommandozeilenargumente überschrieben werden. Wird `DVI` von der `Workbench` gestartet, so verändern auch die `Tool Types` der eventuellen `Workbench`-Argumente, als auch die `Tool Types` von `DVI` selbst das Programmverhalten. Die Einstellung der `Tool Types` erfolgt über die `Workbench`: Bei selektiertem Piktogramm gelangt man über den Menüpunkt `Icons/Information` zum Konfigurationsfenster. Nicht zuletzt bietet sich die Möglichkeit, die Einstellungen über `DVI` Menüpunkte vorzunehmen und

dann in der bereits erwähnten Optionsdatei `dvi.opt` zu speichern.

Im folgenden soll nur die Konfiguration über die Programm-Menüs erläutert werden. Die für die direkten Methoden nötigen Schlüsselworte sind in Tabelle 2.1 aufgeführt und nachfolgend unter den gleichlautenden Menüeinträgen ausreichend erläutert. Für eine weitere, evtl. noch detailiertere Beschreibung kann zusätzlich noch Abschnitt 2.4 über die Kommandozeilenversion von `DVI` zu Rate gezogen werden.

Der Aufruf von `DVI` kann sowohl von der `Workbench` als auch vom `CLI` aus erfolgen. Im `CLI` lautet der Aufruf wie folgt:

```
dvi [optionen] [dvifile[.dvi]]
```

Sowohl die Angabe von Optionen, als auch die Benennung einer zu ladenden `DVI`-Datei sind optional. Für `[optionen]` können eine oder mehrere Angaben vom Format `option=wert` stehen also z.B. `memory=on`.

Von der `Workbench` kann der Name der `DVI`-Datei in der auf dem AMIGA üblichen Art übergeben werden: Bei gedrückter SHIFT-Taste wird zuerst das `DVI`-Programm-Piktogramm einmal angeklickt und dann das Piktogramm der gewünschten `DVI`-Datei doppelt angeklickt. Die `Tool Types` des `DVI`-Programm-Piktogramms und die des `DVI`-Datei-Piktogramms werden von `DVI` als zusätzliche Einstellungen erkannt.

Nach dem Starten von `DVI` werden ein eigener Screen und darauf ein Informationsfenster und eventuell ein Grafikfenster geöffnet. Zusätzlich trägt sich `DVI` in das `Tools`-Menü der `Workbench` ein. Hierdurch wird es ermöglicht, auf der `Workbench` `DVI`-Dateien zu selektieren und durch den Menüeintrag direkt an `DVI` zu übergeben.

Bei aktiviertem `DVI`-Fenster kann man mit der RETURN-Taste, bzw. der BACKSPACE-Taste vorwärts, bzw. rückwärts blättern. Auch ein „Doppelklick“ mit der linken Maustaste blättert eine Seite weiter. Mit den `Cursortasten` kann der aktuelle Seitenausschnitt bestimmt werden. In Anlehnung an die X11/Motif-Version kann auch bei der AMIGA-Version durch Drücken *und Halten* der linken Maustaste der aktuelle Bildausschnitt verschoben werden (deshalb auch hier keine Scrollbalken).

2.3.1 Der Project Titel

Load DVI	A L
Update DVI	A U
Save Options	A S
About	
Quit	A Q

Load DVI Durch Auswahl dieses Menüpunktes oder Betätigung des `Shortcuts` A L, d.h. durch Drücken der beiden Tasten `rechte Amiga-Taste` + L, wird ein Kommunikationsfenster zur Auswahl einer `DVI`-Datei angezeigt. Alternativ kann der Name einer zu ladenden `DVI`-Datei auch beim Aufruf über die

COPIES	=	⟨n⟩	Anzahl Kopien beim Druck
DENSITY	=	⟨dens⟩	Dunkelwert bei Grafiken
EJECT	=	⟨on/off⟩	Seitenvorschub beim Druck
HEIGHT	=	⟨dimen⟩	Seitenhöhe
HMARGIN	=	⟨dimen⟩	Einrückung nach rechts
HOFFSET	=	⟨dimen⟩	Seitenverbreiterung links
HRESOLUTION	=	⟨dimen⟩	Auflösung horizontal
HSPREAD	=	⟨dimen⟩	Seitenverbreiterung rechts
IMGPATH	=	⟨path⟩	Grafiksuchpfad
LANDSCAPE	=	⟨on/off⟩	Querformat
LOGFILE	=	⟨name⟩	Protokolldatei
MAGNIFICATION	=	⟨mag⟩	Vergrößerungsfaktor
MEMORY	=	⟨on/off⟩	`DVI`-Datei im Speicher
OUTPUT	=	⟨printer⟩	Druckertyp
PATH	=	⟨path⟩	Suchpfad für `DVI`-Dateien
PATHMEM	=	⟨n⟩	Speicherplatz für Grafikbefehle
PICTURES	=	⟨on/off⟩	Grafiken anzeigen
PIXMEM	=	⟨n⟩	Speicherplatz für Druck-Bitmap
PKPATH	=	⟨path⟩	Pfad der `PK`-Zeichensätze
REDIRECT	=	⟨name⟩	Druck-Umlenkung in Datei
SEPARATE	=	⟨on/off⟩	Druckqualität
SHOWFONTS	=	⟨on/off⟩	Zeichensatzprotokoll
SINGLESHEET	=	⟨on/off⟩	Einzelblattbetrieb
TFMPATH	=	⟨path⟩	Pfad der `TFM`-Dateien
THINOUT	=	⟨on/off⟩	„Ausdünnen“
TRACECHARS	=	⟨on/off⟩	Zeichenprotokoll
TRACEMEM	=	⟨on/off⟩	Speicherprotokoll
VFPATH	=	⟨path⟩	Pfad für virtuelle `VF`-Zeichensätze
VMARGIN	=	⟨dimen⟩	Einrückung nach unten
VOFFSET	=	⟨dimen⟩	Seitenverlängerung oben
VRESOLUTION	=	⟨dimen⟩	Auflösung vertikal
VSPREAD	=	⟨dimen⟩	Seitenverlängerung unten
WIDTH	=	⟨dimen⟩	Seitenbreite
CALLMF	=	⟨name⟩	Skript zur Zeichensatzerzeugung

Tabelle 2.1: Die Optionen der `DVI` Amigaversion

Kommandozeile, oder beim Start von der Workbench durch `erweiterte Auswahl` angegeben werden.

Update DVI Dieser Menüpunkt bzw. die Tastenkombination A U bewirkt, daß die aktuelle `DVI`-Datei neu geladen und angezeigt wird. Dies kann sinnvoll sein, wenn im Hintergrund eine neue Version der aktuellen `DVI`-Datei erzeugt wurde.

Save Options Über diesen Eintrag bzw. den Shortcut A S können die eingestellten Optionen gespeichert werden. Die Einstellungen werden dabei in einer speziellen Optionsdatei (`dvi.opt`) gespeichert. Diese Datei befindet sich entweder im Ordner `TeX:config`, oder in dem in der Environmentvariablen `TEXCONFIG` gespeicherten Verzeichnis. Die Optionsdatei liegt im `ASCII`-Format vor und kann somit auch von Hand editiert werden.

About Ein wahrlich selbsterklärender Menüeintrag.

Quit Durch Auswahl dieses Menüpunkts oder A Q kann das Programm beendet werden.

2.3.2 Der Screen Titel

Next Page	
Previous Page	
Goto Page	A G
Magnification	A M

Next Page Wird dieser Menüeintrag ausgewählt oder RETURN bzw. ENTER gedrückt, so startet der Aufbau der nächsten Bildschirmseite. Nach mehr oder weniger langer Wartezeit erscheint dann die neue Seite im Fenster.

Previous Page Mit diesem Eintrag oder mit der BACKSPACE-Taste kann man die vorangehende Bildschirmseite aufbauen lassen.

Goto Page Über dieses Menü oder den Shortcut A G gelangt man in ein Kommunikationsfenster, das die Eingabe der gewünschten Bildschirmseite ermöglicht. Wird dieser mit `OK` verlassen, so beginnt `DVI` mit dem Aufbau der gewünschten Bildschirmseite.

Magnification Dieser Eintrag bzw. Drücken der Tastenkombination A M dient zum Einstellen des Vergrößerungsfaktors. Ein Wert von 1000 entspricht Originalgröße, ein Wert von 1200 vergrößert um den Faktor 1.2, usw. Beachten Sie aber bitte, daß die Vergrößerung durch Verwendung größerer Zeichensätze bewirkt wird, die natürlich vorhanden sein müssen. Deshalb sind wohl nur die üblichen Vergrößerungsstufen in Schritten von 1.2^i praktikabel. Ist dieser Wert mit Null belegt, so wird der Wert aus der `DVI`-Datei verwendet.

2.3.3 Der Printer Titel

Print Range	A P
Print Page	
Print Document	

Print Range Über diesen Menüpunkt oder A P gelangt man in ein Fenster, das die

Angabe eines zu druckenden Bereichs ermöglicht. Neben der Start- und Endseite kann auch die Schrittweite eingestellt werden, um so z.B. den Druck der geraden und ungeraden Seiten zu ermöglichen.

Print Page Dieser Menüeintrag ermöglicht den Druck der gerade angezeigten Seite.

Print Document Durch Auswahl dieses Punktes kann das gesamte Dokument gedruckt werden.

2.3.4 Der Setup Titel

Directories
Margins
Options
Printer

Directories Über diesen Eintrag gelangt man in ein Einstellungsfenster, in dem man die gängigen, von `DVI` verwendeten Pfade bequem eintragen kann:

- PK Path: Einstellung der Zeichensatzpfade und -namen. Wie schon einmal bei `DVILW` in Abschnitt 1 erläutert, kann im Zusammenhang mit den Platzhaltern `%d, %h, %m, %s` und `%v` der Name und Pfad der verwendeten Zeichensätze dargestellt werden. Die Platzhalter haben folgende Entsprechungen:

%d	Zeichsatzgröße in dpi
%h	horizontale Geräteauflösung
%m	`MAG___1.2` etc.
%s	Zeichensatzname, z.B. `cmr10`
%v	vertikale Geräteauflösung

  Beispielsweise ergibt also eine Zuweisung wie

  ```
  TeX:pk/%hx%v/%d/%s.%dpk
  ```

  Zeichensatzpfade und -namen im Stil von

  ```
  TeX:pk/100x100/110/cmr10.110pk,
  ```

  die auf dem AMIGA z.B. im Rahmen der `PasTEX`-Implementation Sinn machen.

- TFM Path: Einstellung des Pfades auf die Zeichensatzmetrikdateien, die von `DVI` herangezogen werden, falls die `PK`-Dateien fehlen. Auf diese Weise können wenigstens leere Boxen in der richtigen Breite gesetzt werden.

- IMG Path: Einstellung des Pfades auf die `IMG`-Grafiken.

- GR Path: Einstellung des Pfades auf Dateien, die Grafikbefehle des Treibers enthalten und über `\special{gr input ... }` nachgeladen werden.

- VF Path: Einstellung des Pfades auf die virtuellen Zeichensatzinformationen.
- Call MF: Einstellung des Namens des externen `ARexx`-Scripts, das bei nichtvorhandenen `PK`-Dateien aufgerufen werden soll. Das mitgelieferte Skript `MakePKFont.rexx` versucht den Font zur Laufzeit zu generieren. Ist kein `ARexx`-Skript angegeben, so wird standardmäßig ein Shellskript zur nachträglichen Berechnung der Zeichensätze erzeugt. Zur weiteren Information können noch die Abschnitte 1.3 und 2.6 zu Rate gezogen werden.

Margins Dieser Menüpunkt erlaubt es, in einem eigenen Fenster die von `DVI` benutzten Margins zu setzen:

- H-Offset: Seitenverbreiterung links, additiv zum *hmargin*.
- V-Offset: Seitenverlängerung oben, additiv zum *vmargin*.
- H-Spread: Seitenverbreiterung rechts.
- V-Spread: Seitenverlängerung unten.
- H-Margin: Einrückung der Druckseite nach rechts.
- V-Margin: Einrückung der Druckseite nach unten.
- Width: Seitenbreite. Falls mit Null belegt, wird der Eintrag der `DVI`-Datei übernommen.
- Height: Seitenhöhe. Falls mit Null belegt, wird der Eintrag der `DVI`-Datei übernommen.

Options Bewegt man die Maus über diesen Menüeintrag, so gelangt man in ein Untermenü, durch das die Einstellung einiger weiterer, globaler, Vorgaben für `DVI` ermöglicht wird. Die nachstehenden Punkte lassen sich jeweils ein- oder ausschalten. Der jeweilige Zustand wird dann durch ein gesetztes oder fehlendes Häkchen vor den Menüeinträgen angezeigt.

- Eject: Seitenvorschub nach Druckseite an/aus.
- Landscape: Drehung des Ausdrucks um 90°..
- Memory: Gesamte `DVI`-Datei in den Speicher lesen.
- Pictures: Schalter zum Weglassen der `IMG`-Grafiken.
- Separate: Steuert Druckqualität.
- Showfonts: An- bzw. Ausschalten des Zeichensatzprotokolls.
- Singlesheet: Einzelblattbetrieb oder Endlospapier.
- Thinout: Das „Ausdünnen“der erzeugten Seite wird an-/ausgeschaltet.
- Tracechars: An- bzw. Ausschalten des Zeichenprotokolls.

- Tracemem: An- bzw. Ausschalten des Speicherprotokolls.

Printer Über diesen Menüpunkt gelangt man in ein Fenster, in dem man über sogenannte `Radio Buttons` den gewünschten Drucker einstellen kann:

- None: Es ist kein Drucker angeschlossen.
- NEC P6 low: Ausgabe auf 24-Nadler mit NEC-kompatiblen Befehlssatz in 180×180 dpi.
- NEC P6 high: Ausgabe auf 24-Nadler mit NEC-kompatiblen Befehlssatz in 360×360 dpi.
- NEC P6 mid: Ausgabe auf 24-Nadler mit NEC-kompatiblen Befehlssatz in 360×180 dpi.
- Epson FX-80: Ausgabe auf 9-Nadler mit Epson-kompatiblen Befehlssatz in 240×216 dpi.
- HP Desk-/Laserjet high: Ausgabe auf Laserdrucker mit HP Laserjet II-kompatiblen Befehlssatz in 300×300 dpi.
- HP Desk-/Laserjet low: Ausgabe auf Laserdrucker mit HP Laserjet II-kompatiblen Befehlssatz in 100×100 dpi.
- Canon BJ-300: Ausgabe auf entsprechenden Laserdrucker.

Über zwei weitere `Radio Buttons` kann die Druckausgabe umgelenkt werden:

- Printer: Die Ausgabe soll auf dem ausgewählten Drucker erfolgen. Als Ausgabeschnittstelle verwendet `DVI` die standardmäßig eingestelle Druckerschnittstelle. Diese kann mittels des `Preferences`-Programms `Prefs/Printer` über die `Workbench` eingestellt werden. Die Einstellung ist i.a. bereits bei der Installation der `Workbench` erfolgt.
- File: Die Ausgabe erfolgt in die unmittelbar unterhalb von `File` anzugebende Datei. Diese kann dann später ohne Verwendung von `DVI` ausgegeben werden. Die Ausgabe kann dann auch auf einem anderen Computer, oder sogar auf einem völlig verschiedenen Computersystem erfolgen.

Des weiteren kann die Anzahl der gewünschten Kopien, sowie die Vergrößerungsstufe für Ausdrucke bestimmt werden. (vgl. dazu 2.3.2, Magnification)

2.4 DVI-Kommandozeilenversion

Die Kommandozeilenversion kann mit oder ohne Optionen aufgerufen werden. Bei Aufruf mit Optionen hat die Kommandozeile folgendes Aussehen:

`dvi ⟨name⟩ -i⟨opt⟩ -f⟨xxx:yyy:zzz⟩ -o⟨out⟩ -l⟨log⟩`

⟨name⟩ ist dabei der Name der DVI-Datei, ⟨opt⟩ der Name der Optionendatei, xxx, yyy, zzz sind eine Formatanweisung, out ist ein Ausgabegerät (siehe *output*-Option) und log ist der Name einer Protokolldatei. Zwischen den Schaltern und ihren Parametern darf ein Leerzeichen stehen. Die Reihenfolge der Parameter ist frei und die Angabe optional.

Die Erklärung der Kommandozeilenschalter ist (Groß - und Kleinschreibung erlaubt:

`-i⟨opt⟩` Die DVI-Optionen werden aus der Datei ⟨opt⟩ gelesen. Standardoptionsdateien, d.h. die üblicherweise mitgeliefert werden, sind:

- DVI.OPT
  Diese Optionsdatei wird beim Aufruf standardmäßig geladen, falls sie vorhanden ist und sollte systemspezifische Anpassungen enthalten.
- SCREEN.OPT
  Optionen für Monitoranzeige in variabler Auflösung.
- APOLLO.OPT
  Optionen für Monitoranzeige auf Apollo-Workstations mit etwas größeren Zeichensätzen.
- P6H.OPT
  Optionen für 24-Nadeldrucker mit 360×360 dpi.
- P6M.OPT
  Optionen für 24-Nadeldrucker mit 360×180 dpi.
- P6L.OPT
  Optionen für 24-Nadeldrucker mit 180×180 dpi.
- FX80.OPT
  Optionen für 9-Nadeldrucker mit 240×216 dpi.

Die Erklärung der Optionen erfolgt im nächsten Abschnitt.

`-f⟨xxx⟩`, `-f⟨xxx:yyy⟩`, `-f⟨xxx:yyy:zzz⟩` Die Formatanweisung wählt die zu bearbeitenden Seiten des Dokuments aus. xxx ist die Startseite, yyy die letzte Seite und zzz gibt die Schrittweite an. Die Parameter entsprechen nicht den TEX-Seitennummern, sondern der Reihenfolge der Seiten in der DVI-Datei! Wird die Formatanweisung angegeben, startet DVI sofort mit der Bearbeitung der Seiten, ansonsten wird in den interaktiven Bearbeitungsmodus geschaltet und es erscheint der Prompt DVI>.

`-o⟨out⟩` Das Ausgabegerät wird durch ⟨out⟩ bestimmt. Gültige Angaben sind sind bei der Erklärung der output-Option nachzulesen.

`-l⟨log⟩` Der Treiber protokolliert seine Tätigkeiten in der Datei ⟨log⟩.

Optionen

Die DVI-Optionen werden entweder im interaktiven Modus per Tastatur oder beim Aufruf per Datei mittels `-i⟨name⟩` eingegeben. Bei sichtbarem

Prompt `DVI>` kann eine Optionsdatei auch nachträglich durch Eintippen ihres Namens geladen werden. Die Parameter der Optionen sind *dimen* für gültige TEX-Längenangaben wie 1.1 *cm*, 7 *pt*, 0.3 *in*, usw; *mag* für Vergrößerungsfaktoren mit Standard 1000, Faktor 1.2 = 1200, usw; *dpi* für Geräteauflösungen; *path* für Pfade.

hresolution ⟨*dpi*⟩ Horizontale Auflösung in dpi.

vresolution ⟨*dpi*⟩ Vertikale Auflösung in dpi.

pkpath ⟨*path*⟩ Pfad und Name zum Suchen der `PK`-Dateien. Mehrere Angaben werden — je nach Betriebssystem — durch `;` oder `:` getrennt. Kein Leerzeichen vor und nach dem `;`! Wie schon einmal bei `DVILW` in Abschnitt 1 erläutert, kann im Zusammenhang mit den Platzhaltern `%d, %h, %m, %s` und `%v` der Name und Pfad der verwendeten Zeichensätze dargestellt werden. Die Platzhalter haben folgende Entsprechungen:

%d	Zeichensatzgröße in dpi
%h	horizontale Geräteauflösung
%m	`MAG____1.2` etc.
%s	Zeichensatzname, z.B. `cmr10`
%v	vertikale Geräteauflösung

Beispielsweise ergibt also eine Zuweisung wie

```
pkpath=D:\FONTS\%d\%s.pk
```

Zeichensatzpfade und -namen im Stil von

```
D:\FONTS\111\cmr10.pk.
```

tfmpath ⟨*path*⟩ Pfad zum Suchen der `TFM`-Dateien, die gebraucht werden, wenn eine `PK`-Datei nicht vorhanden ist, um wenigstens leere Boxen der richtigen Breite zu erzeugen, bzw. die bei Verwendung virtueller Zeichensätze gebraucht werden.

vfpath ⟨*path*⟩ Pfad zum Suchen der virtuellen Zeichensatzdefinitionen `.VF`.

imgpath ⟨*path*⟩ Pfad zum Suchen der `IMG`-Dateien. Mehrere Angaben werden durch `;` oder `:` getrennt. Kein Leerzeichen vor und nach dem Trennzeichen!

grpath ⟨*path*⟩ Pfad zum Suchen der nachträglich mit `\special{gr input ... }` einzuladenden Grafikdateien.

path ⟨*path*⟩ Pfad zum Suchen der `DVI`-Dateien. Mehrere Angaben werden durch `;` getrennt. Kein Leerzeichen vor und nach dem `;`!

output ⟨*device*⟩ Gibt Ausgabegerät an. Bisher sind definiert:

- screen
  Monitor mit variabler Auflösung, je nach Computer.
- file
  Datei zur Weiterverarbeitung durch `DISPLAY` (siehe auch Anhang B).

- fx80
  Epson kompatibler 9-Nadler mit 240 × 216 dpi.
- p6low
  NEC P6 kompatibler 24-Nadler mit 180 × 180 dpi.
- p6mid
  NEC P6 kompatibler 24-Nadler mit 360 × 180 dpi.
- p6high
  NEC P6 kompatibler 24-Nadler mit 360 × 360 dpi.
- hplow
  HP Laserjet kompatibler Laserdrucker mit 100 × 100 dpi.
- hphigh
  HP Laserjet kompatibler Laserdrucker mit 300 × 300 dpi.
- bj300
  Canon BJ300 kompatibler Laserdrucker mit 300 × 300 dpi.
- null
  Null-Device zu Testzwecken (Zeichensätze vorhanden, `DVI`-Datei in Ordnung).

format ⟨*list*⟩ Angabe der zu bearbeitenden Seite(n). Die Anwendung entspricht dem `-f` Schalter, nur daß zwischen den Parametern und der Option ein Leerzeichen stehen muß und daß bei Verwendung ohne Parameter die gesamte `DVI`-Datei bearbeitet wird.

lower, upper Ausgabesteuerung über die internen TEX-Seitenzähler `\count0` bis `\count9`.

copies ⟨*n*⟩ Druckt *n* Kopien von jeder Seite.

magnification ⟨*mag*⟩ Seitenvergrößerung. Falls mit Null belegt, wird der Wert aus der `DVI`-Datei verwendet.

width ⟨*dimen*⟩ Seitenbreite. Falls mit Null belegt, wird der Eintrag der `DVI`-Datei übernommen.

height ⟨*dimen*⟩ Seitenhöhe. Falls mit Null belegt, wird der Eintrag der `DVI`-Datei übernommen.

hoffset ⟨*dimen*⟩ Seitenverbreiterung links. Negative Werte mit entgegengesetztem Effekt sind möglich. Parameter wirkt additiv auf *hmargin*.

voffset ⟨*dimen*⟩ Seitenverlängerung oben. Negative Werte mit entgegengesetztem Effekt sind möglich. Parameter wirkt additiv auf *vmargin*.

hmargin ⟨*dimen*⟩ Einrückung der Druckseite nach rechts. Negative Werte mit entgegengesetztem Effekt sind möglich.

vmargin ⟨*dimen*⟩ Einrückung der Druckseite nach unten. Negative Werte mit entgegengesetztem Effekt sind möglich.

hspread ⟨*dimen*⟩ Seitenverbreiterung rechts. Negative Werte mit entgegengesetztem Effekt

sind möglich.

vspread ⟨*dimen*⟩ Seitenverlängerung unten. Negative Werte mit entgegengesetztem Effekt sind möglich.

eject ⟨*on/off*⟩ Seitenvorschub nach Druckseite an/aus.

page [⟨*n*⟩*]* Seitenvorschub am Drucker, ggf. ⟨*n*⟩-fach. Sehr geeignet, um druckerwarmes Toilettenpapier zu produzieren. Für optimale Ergebnisse sollte das Papier aber nicht zu glatt sein.

separate ⟨*on/off*⟩ Steuert Druckqualität. Falls auf `on`, wird die Anzahl der Druckdurchgänge verdoppelt, wobei bei einem Durchgang nur jeweils *nichtbenachbarte* Nadeln drucken.

singlesheet ⟨*on/off*⟩ Umschlaten auf Einzelblattbetrieb mit Wartestellung nach jeder Seite bzw. Endlospapier ohne Warten.

landscape ⟨*on/off*⟩ Drehung des Ausdrucks um 90°. Natürlich darf die Seite nicht zu lang sein und für die Outputroutinen fx80 und p6mid sind spezielle Zeichensätze nötig (216×240 dpi und 180×360 dpi, logisch!).

pictures ⟨*on/off*⟩ Schalter zum Weglassen der `IMG`-Grafiken. An Stelle der Grafiken wird ein leerer Rahmen angezeigt. Dadurch wird der Druckseitenaufbau deutlich schneller!

density ⟨*n*⟩ Faktor zum Einstellen des „Dunkelwertes“ bei `IMG`-Grafiken, der zwischen 0.0 und 1.0 liegen soll. Ein Wert von $n = 1.0$ entspricht der dunkelsten Darstellung.

thinout ⟨*on/off*⟩ Das „Ausdünnen“der erzeugten Seite wird an-/ausgeschaltet.

port ⟨*n*⟩ Setzt auf `IBM PC`-Kompatiblen den Ausgabeport. Gültige Werte liegen zwischen 0 und 3, entsprechend den Ports `LPT0`– `LPT3`.

callmf ⟨*name*⟩ Diese Option existiert nur auf Amiga-Implementationen und erhält den Namen eines `ARexx`-Scripts, das einen nichtvorhandenen Zeichensatz evtl. zur Laufzeit generiert. Es steht das Script `MakePkFont.rexx` zur Verfügung, das im `REXX:`-Verzeichnis stehen muß.

memory ⟨*on/off*⟩ Bei eingeschalteter Option wird die gesamte `DVI`-Datei in den Speicher gelesen und dadurch schneller bearbeitet.

showfonts ⟨*on/off*⟩ An- bzw. Ausschalten des Zeichensatzprotokolls.

tracemem ⟨*on/off*⟩ An- bzw. Ausschalten des Speicherprotokolls.

tracechars ⟨*on/off*⟩ An- bzw. Ausschalten des Zeichenprotokolls. Vorsicht, für jedes einzelne verwendete Zeichen wird angegeben, aus welchem Zeichensatz es stammt und welches Zeichen ausgegeben wurde!

maxmem ⟨*n*⟩ Auf Multitaskingsystemen kann durch Angabe einer maximalen Speichergröße ⟨*n*⟩ Bytes Platz für andere Applikationen freigehalten werden. Ansonsten nimmt sich `DVI` soviel Speicher, als nötig.

pixmem ⟨*n*⟩ Für den Aufbau der Textbitmap werden nur ⟨*n*⟩ Bytes verwendet, anstatt soviel als vorhanden bzw. notwendig. Auf diese Weise kann bei Maschinen mit wenig Speicher bei sehr vielen verwendeten Zeichensätzen durch Beschränkung der Bitmapgröße dennoch ein Ausdruck erzeugt werden, jedoch in mehreren Durchgängen. Mit dieser Variable sollte experimentiert werden, wenn die `DVI`-Meldung „*Out of memory* “ erscheint!

pathmem ⟨*n*⟩ Mit diesem Befehl kann der Speicherplatz für Clipoperationen der Grafikbefehle festgelegt werden. Evtl. kann die Voreinstellung bei komplizierten Operationen zu niedrig sein bzw. kann dieser Speicher für kritische Texte ohne Grafik zum Aufbau der Textbitmap herangezogen werden.

redirect ⟨*name*⟩ Falls mit dieser Variable ein Dateiname angegeben wird, wird nicht direkt auf dem Drucker gedruckt, sondern in die angegebene Datei. Diese Datei kann dann auf einem anderen Rechner ausgedruckt werden, ohne daß ein TEX-System installiert ist.

dvifile ⟨*name*⟩ Neue `DVI`-Datei *name* bearbeiten, ohne `DVI` zu verlassen.

logfile ⟨*name*⟩ Statistiken, wie gewohnt. Per Voreinstellung ausgeschaltet!

input ⟨*name*⟩*, name* Die Optionsdatei ⟨*name*⟩ bzw. ⟨*name.opt*⟩ wird geladen. Entspricht dem Schalter `-i`.

options, *options* ⟨*name*⟩ Ohne Parameter werden die Optionen mit ihren aktuellen Parametern ausgegeben. Mit Parameter wird eine Optionsdatei ⟨`name`⟩ geschrieben.

exit `DVI` verlassen. Der Druckpuffer von 16 KByte wird allerdings noch fertig ausgedruckt.

Sämtliche Optionen können auch abgekürzt verwendet werden, soweit dies möglich ist. Anstelle von „*format*“ kann auch „*f*“ oder anstelle von „*output*“ kann „*out*“ angegeben werden, usw. Einfach probieren!

Environment

Auf PC-kompatiblen Rechnern kann über die Environmentvariable `GRAFX` eine Anpassung an bestimmte Grafikkarten vorgenommen werden, die der Erkennungsalgorithmus nicht automatisch vornehmen kann bzw. kann man andere Grafikmodi einstellen, als die automatische Erkennung. Die allgemeine Syntax des `GRAFX`-Environments ist wie folgt:

```
SET GRAFX=mode,lines,bytes,code[,fgc,bgc]
```

wobei grundsätzlich die ersten vier Parameter angegeben werden müssen, jeweils durch Komma *ohne zusätzliche Leerzeichen* getrennt. Die Bedeutung der Parameter ist wie folgt:

`mode` Der einzustellende Grafikmodus der Grafikkarte in Dezimal.

`lines` Die Anzahl der Bildschirmzeilen im eingestellten Grafikmodus.

`bytes` Die Anzahl Bytes pro Bitplane einer Bildschirmzeile.

`code` Die Auswahl der Funktion, die das Bitmuster einer TEX-Seite in den Bildschirmspeicher kopiert. Dabei stehen folgende Möglichkeiten zur Auswahl (Großschreibung zwingend vorgeschrieben!):

- LOW
  Bildschirmzeilen mit gerader Zeilennummer sind beginnend bei Segmentadresse `B800` abgespeichert, solche mit ungerader Zeilennummer bei Segmentadresse `BA00`. Dies entspricht dem `CGA`-Grafikmodus 6 bei einer Auflösung von 640 × 200 Bildschirmpixeln.

- MID
  Die Bildschirmzeilen sind entsprechend einer modulo 4-Rechnung beginnend bei den Segmentadressen `B800`, `BC00`, `A800` bzw. `AC00` abgespeichert, wobei in jeweils zwei aufeinanderfolgenden Bits ein Bildschirmpixel codiert ist, also vier Farben zur Verfügung stehen. Dies entspricht einem speziellen `CGA`-Grafikmodus 9 bei einer Auflösung von 640 × 400 Bildschirmpixeln, der bei älteren Rechnern vom Typ `NCR PC8` verfügbar ist.

- HIGH
  Die Grafikinformation ist beginnend bei Segmentadresse `A000` in vier Bitplanes abgespeichert. Dies entspricht den hochauflösenden Grafikmodi 16 (`EGA`) bzw. 18 (`VGA`) bei einer Auflösung von 640 × 350 bzw. 640 × 480 Bildschirmpixeln. Das Bitmuster wird durch direkte Programmierung des Grafikkontrollers der Karte übertragen. Dieser Modus funktioniert normalerweise auch bei sog. `Super-VGA`-Karten in den 16-Farb-Grafikmodi.

- MONO
  Die Bildschirmzeilen sind entsprechend einer modulo 4-Rechnung beginnend bei den Segmentadressen `B000`, `B200`, `B400` bzw. `B800` abgespeichert, wobei einem Bit jeweils ein Bildschirmpixel entspricht, also zwei Farben zur Verfügung stehen. Dies entspricht dem Grafikmodus einer `Hercules`-Karte bei einer Auflösung von 720 × 348 Bildschirmpixeln.

`fgc, bgc` Bei den `HIGH`-Modi kann zusätzlich die Farbe der Bildschirmdarstellung gewählt werden. Seit die Treiber Freeware Status haben, ist dies die erste Verbesserung von Benutzerseite. Für die Implementierung und Dokumentation gilt an dieser Stelle Herrn Hans-Joachim Töpfer unserer besonderer Dank! Es bedeuten:

fgc : die Vordergrundfarbe,
bgc: die Hintergrundfarbe.

Die Farben werden entsprechend dem `EGA`-Standard erzeugt. Die Syntax für die Farbeinträge ist folgende:

```
fgc|bgc::=color
color::=number|{LIGHT|BRIGHT|DARK|secondary_color}primary_color
secondary_color|primary_color::=color_name
color_name::=BLACK|BLUE|GREEN|CYAN|RED|MAGENTA|
             BROWN|GRAY|YELLOW|WHITE
```

Die Farbnummern ergeben sich aus einer 6-stelligen Dualzahl `rgbRGB`. `R`, `G`, bzw. `B` aktivieren die Farben Rot, Grün, bzw. Blau in der Primär-Intensität, `r`, `g`, bzw. `b` in der schwächeren Sekundär-Intensität.

Beim EGA-Standard wird folgende Codierung verwendet:

`BLACK`	:	0
BLUE	:	1
GREEN	:	2
CYAN	:	3
RED	:	4
MAGENTA	:	5
BROWN	:	20
GRAY	:	7
YELLOW	:	62
WHITE	:	63

Bei den Farbnamen im `GRAFX`-Eintrag definiert die primary_color die `RGB`-Bits, die secondary_color die `rgb`-Bits.

Die Zusätze haben folgende Bedeutung:

`DARK`	primary_color = `BLACK`;
`BRIGHT`	secondary_color = primary_color;
`LIGHT`	secondary_color = `WHITE`;

Es gibt z.B.

`Red`	(codiert als 4)
`DarkRed`	(codiert als 32)
`BrightRed`	(codiert als 36)
`LightRed`	(codiert als 60)
`WhiteRed`	(codiert als 60)
`RedWhite`	(codiert als 39)

Bei Grafikkarten mit dem `ET4000`-Chip und 1 MB Bildschirmspeicher funktionieren nun auch die Modi mit 1024×768 und 1280×1024 Pixeln.

Die Farbnamen können der besseren Übersichtlichkeit halber mit großen und kleinen Buchstaben (auch gemischt) geschrieben werden.

Tabelle 2.4 gibt einige Beispiele mit kurzer Erklärung. Alle Fälle, die mit *Standard* bezeichnet werden, sollten durch den Grafikkartenerkennungsalgorithmus automatisch eingestellt werden, wobei jeweils der höchstauflösende Modus verwendet wird. Wer jedoch gerne auf seiner `Super-VGA` mit 640×200 Pixeln im `CGA`-Modus arbeitet bzw. arbeiten muß, weil der Monitor nicht mitspielt, der kann seiner Grafikkarte zwangsweise Schonkost verordnen ...

Beachten Sie bitte, daß bei Verwendung des `GRAFX`*-Environments keinerlei Überprüfung der Hardware stattfindet, sondern Sie in eigener Verantwortung handeln. Mit den falschen Parametern können Sie ohne weiteres Ihren Monitor zerstören. Aus diesem Grund müssen Sie eine eventuelle* `GRAFX`*-Einstellung auch explizit bestätigen! Andererseits können Sie*

`SET GRAFX=6,200,80,LOW`	Standard `CGA`
`SET GRAFX=16,350,80,HIGH`	Standard `EGA`
`SET GRAFX=18,480,80,HIGH`	Standard `VGA`
`SET GRAFX=135,348,90,MONO`	Standard `Hercules`
`SET GRAFX=?,600,100,HIGH`	`Super-VGA`
`SET GRAFX=41,600,100,HIGH,BLACK,WHITE`	`ET4000` 800×600 Schwarz auf Weiß
`SET GRAFX=55,768,128,HIGH,GREEN,BLACK`	`ET4000` 1024×768 Grün-Monochrom
`SET GRAFX=61,1024,160,HIGH`	`ET4000` 1280×1024

Abbildung 2.4: Beispiele für `GRAFX`-Environments

natürlich kleine Unzulänglichkeiten bei der automatischen Grafikkartenerkennung durch Setzen dieser Environmentvariablen ausgleichen. So werden manche `Hercules`*-Karten nur bei jedem zweiten* `DVI`*-Durchlauf als solche erkannt. In diesem Fall sollte man auf* `GRAFX` *zurückgreifen. Gleiches gilt natürlich für spezielle Grafikmodi exotischer Karten.*

2.5 Arbeitsweise von `DVI`

Nachdem alle Optionen eingegeben wurden, wird die Bearbeitung durch eine „*format*“ bzw. „Start Print“-Anweisung gestartet. Der Speicherbedarf für eine Seite wird berechnet, die Zeichensätze geladen und die erste Seite (evtl. in mehreren Durchgängen) im Speicher aufgebaut und angezeigt/gedruckt.

Bei der Bildschirmanzeige wird evtl. nur ein Ausschnitt der Seite angezeigt. Die nichtsichtbaren Teile können durch Scrollbalken oder der Cursor-Tasten (alle Rechner) in das sichtbare Fenster verschoben werden. Hat man sich an der Bildschirmseite sattgesehen, kann man durch Drücken von `N` die nächste Seite zur Anzeige bringen (oder den nächsten Durchgang für dieselbe Seite starten) bzw. durch Drücken von `Q` die Rückkehr in die interaktive Optionseingabe erzwingen. Wird bei der letzten Seite der Formatangabe versucht, eine weitere Seite des Dokumentes zur Anzeige zu bringen, kehrt man trotzdem zur Optionseingabe zurück.

2.6 Besonderheiten

Falls Zeichensätze fehlen, wird eine Datei `missing.fnt` bzw. `MISSING.BAT` zur automatischen Generierung erzeugt, die auf Atari-Rechnern zur TEXshell des Lindner-TEX kompatibel ist und auf `UNIX`-Rechnern sowie AMIGAs ein Shell-Script ist, das nach Eingabe von `./missing.fnt <driver>` auf `UNIX`-Maschinen bzw. `execute missing.fnt <driver>` auf dem AMIGA die Zeichensätze automatisch erzeugt. `<driver>` steht hierbei für den METAFONT-Namen des entsprechenden Ausgabegerätes (`atari_screen, OneZeroZero, apollo_screen`, etc.). Auf PC-kompatiblen Rechnern werden die Zeichensätze durch den Batchjob über weitere spezielle Batchjobs automatisch erzeugt.

Auf dem Amiga kann alternativ ein in der Konfigurationsvariablen `CALLMF` angegebenes externes `ARexx`-Programm gestartet werden, das dann entweder den Zeichensatz generiert, oder eine entsprechende Batchdatei erzeugt/erweitert. Im Fall von `MakePKFont.rexx` wird versucht, den fehlenden Font direkt zu erzeugen. Diese Vorgehensweise ist jedoch sehr Speicheraufwendig — neben der `DVI`-Datei muß gleichzeitig noch METAFONT samt Daten im Hauptspeicher platz finden — weshalb man bei Systemkonfigurationen mit bis zu 2 MB eventuell *nicht* auf diese Methode zurückgreifen kann.

Falls Zeichensätze fehlen, erfolgt eine Abfrage, ob die Bearbeitung abgebrochen werden soll, oder ob mit leeren Bitmustern für die fehlenden Buchstaben gearbeitet werden soll.

Falls Sie Zeichensätze mit `missing.fnt` auf `UNIX`-Geräten erzeugt haben, sollten Sie die entstandenen Dateien, die auf `pk` enden, in den entsprechenden Ordner

`/usr/local/lib/tex/pk101` bzw.
`/usr/local/lib/tex/pk120`

kopieren, um diese Zeichensätze permanent zur Verfügung zu haben.

Auf Amiga-Computern werden die erzeugten Zeichensätze direkt im ersten in der Konfigurationsvariablen `PKPATH` angegebenen Verzeichnis abgelegt. Die durch METAFONT erzeugten temporären Dateien werden hier automatisch gelöscht.

Auf allen Computern ist es sinnvoll, die Temporärdateien der Zeichensatzerzeugung, nämlich die auf `gf` endenden Dateien und Dateien mit Endung `.log` und `.tfm` zu löschen, falls dies nicht schon, wie auf dem Amiga, geschehen ist.

Natürlich kann jede beliebige Auflösung am Bildschirm dargestellt werden, wenn nur die Zeichensätze vorhanden sind. Wird z.B. `DVI` aufgerufen mit

`DVI -Ip6h`,

wird das Programm mit den Standardoptionen für die Druckausgabe an 24-Nadlern mit 360×360 dpi geladen und es erscheint der Prompt `DVI>`. Werden nun als weitere Optionen `out screen`, sowie `hres=360` und `vres=360` angegeben, wird nicht gedruckt, sondern alles in der Auflösung 360×360 dpi am Bildschirm angezeigt. Bei z.B. der `GEM`-Version reicht das Neusetzen des `PK`-Pfades und das Einstellen der Auflösung (H-Resolution/V-Resolution), um dasselbe Ergebnis zu erzielen. Natürlich sind wegen der Größe des Bitmusters wahrscheinlich — abhängig vom Speicherausbau Ihres Computers — mehrere Durchgänge pro Seite erforderlich.

2.7 Installation

Zur Installation müssen lediglich die entsprechende Datei — `dvi` bei `UNIX` und Amiga, `dvi.prg` bei Atari ST oder `dvi.exe` bei `MS-DOS` — in einen Ordner kopiert werden, der im Systemsuchpfad enthalten ist und einige Pfade

und Optionen richtig eingestellt werden. Will man DVI auf dem AMIGA nur von der Workbench aus benutzen, so muß DVI nichteinmal im Pfad stehen. Bei den GEM- und Motif-Versionen müssen außerdem alle Optionsdateien .DVO gelöscht und neu erzeugt werden, da bei neu hinzugekommenen Optionen die Binärformate der Dateien inkompatibel sind. Als Benutzer merken Sie allerdings von diesen Inkompatibilitäten erst etwas, wenn der Treiber sich seltsam verhält. Bitte Vorsicht! Für die AMIGA- und die Kommandozeilenersion ist die Klartextdatei dvi.opt entsprechend anzupassen. Beim AMIGA muß diese im Ordner TeX:config liegen. Alternativ kann mit der Environmentvariablen TEXCONFIG ein anderes Verzeichnis gewählt werden. Auf MS-DOS Systemen muß die Datei dvi.opt in einen Ordner kopiert werden, der im Suchpfad enthalten ist.

Für den AMIGA wird es in der näheren Zukunft ein Installationsscript für den Installer von Commodore geben, mit dem dann die oben beschriebenen Aktionen nicht mehr zu Fuß erledigt werden müssen.

Zum Arbeiten mit reinen Texten ohne Grafik muß der Name und Pfad der Zeichensätze entsprechend der Organisation der Zeichensatzdateien auf der Festplatte eingestellt werden. Bei UNIX ist per Voreinstellung

```
./%s.%dpk.:/usr/local/lib/tex/pk%h/%s.%dpk
```

eingebaut, wobei alle Auflösungsstufen eines Zeichensatzes in einem dieser beiden Ordner liegen. Bei Systemen mit 8 Zeichen langen Dateinamen und 3 Zeichen langen Erweiterungen sind die Auflösungsstufen der Zeichensätze naturgemäß nicht im Dateinamen unterzubringen. Man hilft sich dahingehend, daß die Zeichensatznamen nur aus dem Basisnamen bestehen — z.B. cmr10 — und die Endung .pk erhalten. Die Auflösungsstufen der Zeichensätze werden auf eine Ordnerhierarchie abgebildet, wobei die Ordnernamen die Vergrößerungsstufe — z.B. mag____1.000 — oder die Auflösung in dpi — z.B. 101 — bezeichnen. Diese Auflösungsstufe errechnet sich zu Gerätegrundauflösung * Vergrößerungsstufe, hier also 101 dpi * 1.000. Außerdem muß diese Gerätegrundauflösung über die Optionen hres und vres eingestellt werden.

Sollen schließlich noch Fremdgrafiken eingebunden werden, die sich nicht im aktuellen Arbeitsordner befinden, sind die Pfade PS und IMG entsprechend zu setzen. Gleiches gilt bei eingebauten Grafikbefehlen, die über externe Dateien mittels \special{gr input ... } eingeladen werden.

Kapitel 3. Virtuelle Zeichensätze

Beginnend mit Version 3.0 der Treiber sind `DVI` und `DVILW` in der Lage, virtuelle Zeichensätze zu verarbeiten. Damit steht eine einfach zu bedienende und elegante Möglichkeit zur Verfügung, verschiedene Zeichensatzbeschreibungen mit `DVI`-Treibern zu verwenden, ohne beim Schreiben mit TEX auf spezielle Macros zurückgreifen zu müssen.

Bei virtuellen Zeichensätzen handelt es sich um ein Interface zwischen der Codierung der Zeichen, wie sie TEX handhabt und der Art und Weise, wie das Ausgabegerät dieses Zeichen schließlich darstellt. Im einfachsten Fall handelt es sich also nur um eine Zwischenstufe in der Zeichensatzbearbeitung, die bei Übersetzung der `.dvi`-Datei die Umsetzung des Codes des aktuellen Zeichens in den entsprecheden Code der `.tfm`-Datei vornimmt. In nicht so einfachen Fällen können in der Beschreibung virtueller Zeichensätze auch rekursive Aufrufe von Zeichensatzbeschreibungen, Linienkommandos oder sogar `\special`-Kommandos für den `DVI`-Treiber stehen. Eine Spezifikation der Beschreibungssprache `VPL` kann man in den TEXhax 1990, Issue 11–13 oder in der TUGboat v. 11, no. 1 vom April 1990, Seite 13–23 finden. Die Programme zur Bearbeitung der `VPL`-Sprache — `VFTOVP` und `VPTOVF` — gibt es ja schon seit einiger Zeit mit jeder ordentlichen TEX-Implementierung.

Zur Motivation — nach dem Motto „Brauche ich das denn?“— stellen sie sich folgende Situation vor: Sie haben einen PostScript-Drucker, dazu teuer gekaufte Zeichensätze wie `Garamond` oder `Lucida` und ein TEX-System mit einem Treiber, der *keine* virtuellen Zeichensätze versteht. Im günstigsten Fall hat dann jemand per Hand bzw. mit `PLTOTF` `TFM`-Dateien gestrickt und Sie können Ihren Text mit den kostbaren Zeichensätzen schreiben, solange Sie keine Zeichen wie „ß“ oder „Ä“ verwenden, die von TEX und PostScript unterschiedlich codiert werden. Im Übrigen fehlt Ihnen eine Preview-Möglichkeit, sofern Sie nicht ein Display-PostScript-System zur Verfügung haben. Jeder, der schon einmal mit TEX gearbeitet hat, weiß, wie unbefriedigend und papierfressend eine solche Arbeitsweise wäre ...

Das erste uns bekannte Treiberprogramm, das in diesem Zusammenhang Abhilfe schaffte, war `DVIPS` von Tomas Rokicki. Durch Verwendung virtueller Zeichensätze war jetzt immerhin die Verwendung von Sonderzeichen möglich. Zudem wird mit diesem Treiber ein Programm (`AFM2TFM`) mitgeliefert, das die Zeichensatzmetrikbeschreibung und die Zeichensatzcodierung der PostScript-Zeichensätze in den `.AFM`-Dateien in einen virtuellen Zeichensatz und eine zugehörige `.TFM`-Datei übersetzt. Fehlt also nur noch ein Preview ...

Basierend auf der Arbeit von Rokicki und Knuth wurde für `DVILW` *und* `DVI` die Verwendung virtueller Zeichensätze implementiert. Damit ist ein weiterer Schritt in Richtung Benutzerfreundlichkeit getan, denn jetzt braucht

man nur noch entsprechende virtuelle Zeichensätze, die die Umsetzung der PostScript-Zeichensätze auf die Computer Modern Zeichensätze erledigen, um mit Bildschirmpreview und Druckerausgabe mit PostScript-Zeichensätzen zu arbeiten. Es wird übrigens noch ein Freiwilliger für die Anpassung von AFM2TFM gesucht!

Da uns momentan die Verwendung von PostScript-Zeichensätzen als sinnvollste Anwendung virtueller Zeichensätze erscheint und Tomas Rokicki auf diesem Gebiet Pionierarbeit geleistet hat, orientiert sich der Rest dieses Abschnittes eng an die Anleitung zu DVIPS und AFM2TFM bzw. ist sogar eine direkte Übersetzung. Warum das Rad neu erfinden?

3.1 AFM2TFM

Damit DVI und DVILW mit virtuellen Zeichensätzen arbeiten können, müssen zunächst die zugehörigen .vf und ggf. die zugehörigen .tfm-Dateien vorhanden sein. Für PostScript-Zeichensätze liegt die benötigte Information in Form von .afm-Dateien vor, die mittels AFM2TFM in die entsprechenden .vf und .tfm-Dateien umgesetzt wird. Angenommen, Sie wollen mit dem Palatino-Roman Zeichensatz in TEX arbeiten, dann müssen Sie zunächst die von TEX und DVI bzw. DVILW benötigten Zeichensatzinformationen mit dem Kommando

```
afm2tfm Palatino-Roman -v pplr rpplr
```

erzeugen. Im folgenden wird übrigens von einem Betriebssystem ausgegangen, das zwischen Klein- und Großschreibung unterscheidet und beliebige Zeichen und Dateinamenslängen verkraftet. Für Systeme mit z.B. 8 + 3-Beschränkungen wie MS-DOS lesen Sie bitte Abschnitt 3.2. Die virtuelle Zeichensatzinformation in der VPL-Sprache muß dann noch mittels

```
vptovf pplr.vpl pplr.vf pplr.tfm
```

in die für den DVI-Treiber verständliche Binärform übersetzt und eine für TEX benötigte .TFM-Datei erzeugt werden. Die gerade erzeugten Dateien pplr.vf,pplr.tfm und rpplr.tfm müssen dann noch in die Unterverzeichnisse kopiert werden, wo sie von TEX bzw. DVI/DVILW erwartet werden (Je nach Betriebssystem verschieden. Bitte Installationsanleitungen lesen). Jetzt können Sie den Palatino-Roman Zeichensatz in TEX mit

```
\font\myfont=pplr at 11pt
{\myfont Dies ist jetzt die PostScript-Schrift
Palatino-Roman mit Umlauten wie "A, Sonderzeichen
wie "s und Ligaturen!}
```

verwenden und erhalten folgenden Ausdruck:

Dies ist jetzt die PostScript-Schrift Palatino-Roman
mit Umlauten wie Ä, Sonderzeichen wie ß und Ligaturen!

→ 3.2

Bitte beachten Sie, daß eigentlich zwei Varianten der Palatino-Roman Schrift für TEX existieren, nämlich die Eins-zu-Eins-Umsetzung ohne spezielle Satzinformation `rpplr` und der virtuelle Zeichensatz `pplr` mit der Umsetzung für Umlaute, Ligaturen, etc.

Um das Dokument mit diesem Zeichensatz schließlich zu drucken, muß noch in der Datei `DVILW.MAP` ein Eintrag stehen/nachgetragen werden, der die Beziehung zwischen dem PostScript-Namen Palatino-Roman und der Eins-zu-Eins-Font-Metrik `rpplr.tfm` herstellt. Es ist übrigens wichtig, die Eins-zu-Eins-Umsetzung anzugeben, da die Umsetzung der Zeichencodes in der `.dvi`-Datei durch den virtuellen Zeichensatzmechanismus in genau diese Eins-zu-Eins-Umsetzung erfolgt! Die Einträge in `DVILW.MAP` haben folgende Form:

```
rpplr = Palatino-Roman
```

Der erste Eintrag ist der Namen der `.TFM`-Datei der Eins-zu-Eins-Umsetzung ohne Extension, der zweite der Namen des Zeichensatzes, wie er mit dem `findfont`-PostScript-Operator gefunden werden kann (ohne führenden „/“).

3.2 Namenskonventionen für Zeichensätze

Im vorigen Abschnitt haben Sie schon die Datei `DVILW.MAP` kennengelernt. Die beiden Einträge jeder Zeile wurden als Dateiname einer TEX-`TFM`-Datei und als Name eines PostScript-Zeichensatzes vorgestellt. Vielleicht haben Sie sich inzwischen schon über den etwas kryptisch anmutenden Namen des ersten Eintrags gewundert, doch hinter dieser willkürlich anmutenden Buchstabenkombination steht ein ausgklügeltes Namenssystem, das Benutzern antiquierter Betriebssysteme wie `MS-DOS` erlaubt, mit den Beschränkungen der Dateinamenslänge trotzdem eine Fülle von Informationen zu codieren. Mit anderen Worten: Der erste Eintrag ist ein tatsächlicher Dateiname, der auch als Alias für den tatsächlichen (und längeren) PostScript-Namen dient.

Die Initiatoren dieses Namenssystems waren — wieder einmal — Mittelbach und Schöpf, deren Artikel in der TUGboat, v. 11, no. 2 vom Juni 1990 Karl Berry zu der vorliegenden Namenskonvention inspiriert hat.

Die acht Buchstaben des Basisdateinamens sind folgendermaßen aufgeteilt:

```
FTTWVEDD
```

wobei die einzelnen Buchstaben folgende Information codieren (Liste der Abkürzungen für PostScript folgt):

`F` Foundry. Hersteller des Zeichensatzes. Wird ggf. weggelassen.

`TT` Typeface name. Name des Zeichensatzes.

`W` Weight. Die Dicke der einzelnen Zeichen.

`V` Variant. Verschiedene Ausprägungen desselben Zeichensatzes wie z.B. *Italic* oder SMALL CAPS. Die Ausprägung wird weggelassen, falls sie und der folgende Parameter Expansion „normal“ sind.

`E` Expansion. Die Breite der Zeichen. Wird weggelassen, falls „normal“.

`DD` Design size. Punktgröße der Zeichen. Wird weggelassen, falls alle Zeichensatzgrößen durch Skalieren aus einer einzigen `TFM`-Datei hervorgehen.

3.2.1 Foundry

Die Liste der Hersteller von Zeichensätzen:

a	Autologic
b	Bitstream
c	Agfa-Compugraphic
g	Free Software Foundation (`g` for GNU)
h	Bigelow & Holmes (with apologies to Chuck)
i	International Typeface Corporation
p	Adobe (`p` for PostScript)
r	reserved for use with virtual fonts; see below
s	Sun

3.2.2 Typeface Families

Die Liste der Zeichensatznamen:

ad	Adobe Garamond	go	Goudy Oldstyle
ag	Avant Garde	gs	Gill Sans
ao	Antique Olive	jo	Joanna
at	American Typewriter	lc	Lucida
bb	Bembo	lt	Lutetia
bd	Bodoni	nc	New Century Schoolbook
bg	Benguiat	op	Optima
bk	Bookman	pl	Palatino
bl	Balloon	pp	Perpetua
bv	Baskerville	rw	Rockwell
bw	Broadway	st	Stone
cb	Cooper Black	sy	Symbol
cl	Cloister	tm	Times
cr	Courier	un	Univers
cn	Century	uy	University
cs	Century Schoolbook	zc	Zapf Chancery
hv	Helvetica	zd	Zapf Dingbats
gm	Garamond		

3.2.3 Weight

Die Liste der Zeichendicken, grob geordnet nach Dicke (zuerst die feinen Schriftdicken):

a	hairline	d	demi
t	thin	s	semi
i	extra light	b	bold
l	light	x	extra bold
k	book	h	heavy
r	regular	c	black
m	medium	u	ultra

3.2.4 Variants

Die Liste der Ausprägungen:

a	alternate	n	informal
b	bright	o	oblique (z.B. slanted)
c	small caps	r	normal (roman oder sans)
e	engraved	s	sans serif
g	grooved (wie beim `IBM` logo)	t	typewriter
h	shadow	u	unslanted italic
i	(text) italic	x	expert
l	outline		

Es gibt einige Ausnahmen bei der Namensbildung mit Ausprägungen. Wenn die Ausprägung „r“ und die Breite der Zeichen (Expansion) „normal“ ist, dann werden beide weggelassen. Falls die normale Ausprägung einer Schrift serifenlos ist, wie z.B. bei Helvetica, sollte als Ausprägung „n“ an Stelle von „s“ benutzt werden. Die „alternate“ und „expert“-Ausprägungen werden von einigen PostScript-Zeichensätzen mit speziellen Zeichen und zusätzlichen Ligaturen benutzt.

3.2.5 Expansion

Die Liste der Zeichenbreiten, geordnet von schmalen zu breiteren Versionen:

o	extra condensed	x	extended (per Hand)
c	condensed (by hand)	e	expanded (automatisch)
n	narrow (automatisch)	w	wide
r	regular, normal, medium (normalerweise weggelassen)		

Die Zeichenbreiten können in PostScript automatisch, wie z.B. durch den `scale`-Operator, als auch per Hand für jeden Buchstaben einzeln codiert werden. Deshalb die unterschiedlichen Namen.

3.3 Namenskonventionen für virtuelle Zeichensätze

Wenn Sie sich an den vorvorigen Abschnitt erinnern, können Sie sich das Namensschema für virtuelle (PostScript-)Zeichensätze sicher schon denken, denn der Name des virtuellen Zeichensatzes sollte nach Karl Berry's Schema aufgebaut sein.

Bei PostScript-Zeichensätzen kam noch das Problem der verschiedenen Codierung einzelner Zeichen hinzu, das durch Verwendung zweier `.TFM`-Dateien gehandhabt wurde, wobei die Eins-zu-Eins-Codierung ein „r“ als

Prefix erhielt, ansonsten aber der Name dem Schema entsprach. Für reine Anwender ist durch die Verwendung der `.TFM`-Datei des virtuellen Zeichensatzes ohnehin keinerlei Abweichung vom Namensschema vorhanden. Es folgt noch eine Liste der Namen der *virtuellen* Zeichensatznamen der gebräuchlichsten PostScript-Zeichensätze:

pagk	AvantGarde-Book	pncri	NewCenturySchlbk-Italic
pagkc	AvantGarde-Book	pncr	NewCenturySchlbk
pagko	AvantGarde-BookOblique	pncrc	NewCenturySchlbk
pagd	AvantGarde-Demi	pplb	Palatino-Bold
pagdo	AvantGarde-DemiOblique	pplbi	Palatino-BoldItalic
pbkd	Bookman-Demi	pplbu	Palatino-BoldUnslanted
pbkdi	Bookman-DemiItalic	pplrrn	Palatino-Narrow
pbkl	Bookman-Light	pplrre	Palatino-Expanded
pbkli	Bookman-LightItalic	pplri	Palatino-Italic
pbklc	Bookman-Light	pplr	Palatino
pcrb	Courier-Bold	pplro	Palatino-Oblique
pcrbo	Courier-BoldOblique	pplru	Palatino-Unslanted
pcrro	Courier-Oblique	pplrc	Palatino
pcrr	Courier	psyr	Symbol
phvb	Helvetica-Bold	psyro	Symbol-Oblique
phvbo	Helvetica-BoldOblique	ptmb	Times-Bold
phvro	Helvetica-Oblique	ptmbi	Times-BoldItalic
phvr	Helvetica	ptmrrn	Times-Narrow
phvrc	Helvetica	ptmrre	Times-Expanded
phvbrn	Helvetica-Narrow-Bold	ptmri	Times-Italic
phvbon	Helvetica-Narrow-BoldOblique	ptmro	Times-Oblique
phvron	Helvetica-Narrow-Oblique	ptmr	Times-Roman
phvrrn	Helvetica-Narrow	ptmrc	Times-Roman
pncb	NewCenturySchlbk-Bold	pzcmi	ZapfChancery-MediumItalic
pncbi	NewCenturySchlbk-BoldItalic	pzdr	ZapfDingbats

Für Kommentare oder Erweiterungen des Namensschemas können Sie sich direkt an Karl Berry wenden:

`karl@cs.umb.edu`

`135 Center Hill Road`
`Plymouth, MA 02360`

Kapitel 4. Die Grafikbefehle von DVI und DVILW

Als letzter Aspekt der Treiber bleibt noch die Erklärung der Grafikmöglichkeiten. DVI und DVILW zeichnen sich hier durch besondere Fähigkeiten aus. Sowohl zwei Fremdgrafikformate, als auch ein Satz von eingebauten Befehlen werden unterstützt. Um die Kompatibilität des TEX-Eingabetextes zu wahren, werden alle Grafikbefehle von TEX mittels \special-Befehlen angesteuert, auch die eingebauten Befehle.

4.1 Specials

! → Um unnötige Wiederholungen zu vermeiden, wird im folgenden jeweils nur der \special-Text beschrieben, obwohl in der Eingabedatei jeweils

\special{⟨*befehl*⟩ ⟨*argumente*⟩}

stehen muß, um die korrekte TEX-Syntax zu erhalten.

! → Beachten Sie bitte unbedingt die Klein-/Großschreibung der Befehle, da die Treiber bei unbekannten Specials keine Warnung ausgeben und bei eigenen Befehlen Klein-/Großschreibung beachten! Falls Sie also keine Grafikausgabe erhalten, sollten Sie einmal die genaue Syntax beachten.

4.1.1 Fremdgrafikformate

Zwei Fremdformate werden unterstützt. Zum einen die IMG-Grafiken von GEM, zum anderen POSTSCRIPT.

graphic img ⟨*filename*⟩

An der aktuellen Seitenposition wird die in der Datei ⟨*filename*⟩ gespeicherte IMG-Grafik eingefügt. Die linke obere Ecke der Grafik kommt dabei am sog. „current point“ zu liegen. Der „current point“ wird nicht verändert. Der
→ 4.2 Benutzer hat selbst von TEX aus für Platz für die Grafik zu sorgen.

ps ⟨*filename*⟩

An der aktuellen Position im POSTSCRIPT-Code wird der Inhalt der Datei ⟨*filename*⟩ eingefügt. Naturgemäß funktioniert dieser Befehl nur mit DVILW, da DVI (vorerst) keinen eingebauten POSTSCRIPT Interpreter hat. Es liegt
→ 4.2 vollkommen in Ihrer Hand, was Sie in die Datei schreiben. Sie sollten aber auf den Aufbau der TEX-Seite achten, d.h. mit \hbox- und \vbox-Befehlen genügend Platz für die Grafik reservieren. Der Inhalt der Datei wird übrigens in ein gsave/grestore-Paar eingebunden und der showpage-Operator lahmgelegt, so daß der TEX-Teil der Ausgabeseite nicht manipuliert werden kann.

postscript ⟨*commands*⟩

Die PostScript-Befehle aus ⟨*commands*⟩ werden in das PostScript-Programm eingefügt, das die aktuelle Druckseite beschreibt. Sehr geeignet für Unsinn jeglicher Art! Auch dieser Befehl ist nur unter `DVILW` verfügbar. Ein Anwendungsbeispiel für diesen Befehl ist folgender TEX/LATEX-Code:

```
\vbox to 100bp{\vss % a bp is the same as a PostScript point
\special{postscript newpath 0 0 moveto 100 100 lineto
354 0 lineto closepath gsave 0.8 setgray fill grestore
stroke}}
```

Das Ergebnis ist Abbildung 4.1. Das Beispiel stammt übrigens aus der Anleitung von `DVIPS` von Tomas Rokicki. Die Befehle werden übrigens zur Sicherheit in ein `gsave`/`grestore`-Paar eingebunden.

Abbildung 4.1: Anwendungsbeispiel für das *postscript*-Special

¨ ⟨*commands*⟩

Bedeutung wie *postscript* ⟨*commands*⟩ und nur aus Kompatibilitätsgründen zu `DVIPS` von Tomas Rokicki vorhanden.

ps: ⟨*commands*⟩

Die PostScript-Befehle aus ⟨*commands*⟩ werden in das PostScript-Programm eingefügt, das die aktuelle Druckseite beschreibt. Sehr geeignet für Unsinn jeglicher Art! Auch dieser Befehl ist nur unter `DVILW` verfügbar. Die PostScript-Befehle werden übrigens *nicht* zur Sicherheit in `gsave` und `grestore` eingebunden, so daß Sie wirklich Unsinn anstellen können. Die Anzahl der Doppelpunkte im *ps:*-Kommando ist übrigens — wiederum aus Kompatibilitätsgründen — freigestellt.

In Bild 4.2 folgt noch ein Beispiel für dieses Special, das wiederum der Anleitung von `DVIPS` von Tomas Rokicki entnommen wurde. Wie im Original wird auch hier auf jeglichen Kommentar verzichtet!

```
\def\rotninety{\special{ps:currentpoint currentpoint
translate 90 rotate neg exch neg exch translate}}
\font\huge=cmbx10 at 14.4truept
\setbox0=\hbox to0pt{\huge A\hss}\vskip16truept
\centerline{\copy0\special{ps:gsave}\rotninety\copy0
\rotninety\copy0\rotninety\box0\special{ps:grestore}}
\vskip16truept
```

Abbildung 4.2: Spaß mit PostScript

PSFile=⟨filename⟩ llx=⟨llx⟩ lly=⟨lly⟩ urx=⟨urx⟩ ury=⟨ury⟩ rwi=⟨rwi⟩

Ähnlich dem *ps*-Special wird an der aktuellen Position im PostScript-Code der Inhalt der Datei ⟨*filename*⟩ eingefügt. Allerdings muß über eine Reihe weiterer Parameter die Bounding Box des Bildes und die tatsächliche Breite des Bildes in PostScript-Punkten angegeben werden. Aus diesen Angaben wird dann das Bild richtig skaliert und positioniert. Direkte Verwendung dieses Special erscheint nicht sinnvoll, da über die `EPSF.STY`-Macros für TEX und LATEX die Verwendung automatisiert ist. Lesen Sie hierfür bitte Abschnitt 4.2.3.

Naturgemäß funktioniert das *PSFile*-Special nur mit `DVILW`, da `DVI` (vorerst) keinen eingebauten PostScript Interpreter hat.

4.1.2 Eingebaute Grafikbefehle

Die eingebauten Grafikbefehle werden alle durch `gr` eingeleitet. Danach kann eine Liste von verschiedenen Grafikbefehlen kommen, die jeweils durch `;` getrennt werden müssen. Natürlich kann man auch jeden Befehl einzeln mit `\special{gr ...}` abschicken.

! → Die direkte Angabe der `\special`-Befehle, so wie im Tutorial aus didaktischen Gründen geschehen, führt zu Problemen mit dem `\put`-Befehl (siehe dazu Abschnitt 4.2.2). Es wird dringendst empfohlen, die `BILDMAC`-Macros zu verwenden!

! → Nachfolgend erklärte Befehle erwarten meistens Parameter eines bestimmten Typs. Es werden die Parameterbezeichnungen *real* für Gleitpunktzahlen und *dimen* für Längenangaben in TEX-ähnlicher-Syntax (wie z.B. -5.33 *cm* oder 5.1 *pt*) bzw. ebenfalls Gleitpunkzahlen verwendet. Im Unterschied zu TEX benötigen die Treiber zum korrekten Erkennen der Bemaßungseinheiten Leerzeichen zwischen syntaktischen Einheiten. So ist z.B. in TEX die Angabe von `1 truecm` zulässig, muß aber in der Treibersyntax als `1 true cm` geschrieben werden. Andererseits verstehen die Treiber als Zahlenwerte bei Längenangaben Ausdrücke aus den vier Grundrechenarten und Klammern, während TEX wirklich eine Zahl erwartet.

! → Alle Einstellungen der Grafikparameter sind nur auf der Druckseite gültig, auf der sie verändert wurden. Durch einen Seitenumbruch werden alle Parameter wie durch ein `setdefaults`-Kommando zurückgesetzt.

%

Wie in TEX dient das %-Zeichen dazu, den Rest der Eingabezeile als Kommentar zu markieren.

def

Das Schlüsselwort `def` dient zum Definieren von Variablen und Funktionen. Die Syntax einer Variablendefinition/-änderung ist wie folgt:

`def ⟨identifier⟩=⟨expression⟩;`

wobei `identifier` ein beliebiger Name ist. Falls der Name schon vorhanden sein sollte, wird der bisherige Wert durch die neue Definition überschrieben.

Eine Funktionsdefinition sieht wie folgt aus:

`def ⟨identifier⟩(⟨p1⟩,...,⟨pn⟩)=⟨expression⟩;`

wobei `identifier` ein noch nicht benutzter Name ist. Die Parameter `p1` bis `pn` dürfen schon als Variablen benutzt sein. In diesem Fall wird aber eine lokale Kopie beim definierenden Ausdruck `expression` verwendet, so daß die Variable nicht verändert wird und auch nicht zur Definition herangezogen werden kann.

Der definierende Ausdruck `expression` hat in beiden Fällen folgenden rekursiven Aufbau:

```
<expression>::=<expression>+<expression> |
               <expression>-<expression> |
               <expression>*<expression> |
               <expression>/<expression> |
               <factor>

<factor>::=<real number> |
           <identifier> |
           <function evaluation> |
           (<expression>)

<function evaluation>::=<identifier(<exp1>,...,<expn>)
```

Zahlreiche Variablen und Funktionen sind schon vordefiniert. Bitte informieren Sie sich durch Abbildung A.14.

Tx,Ty

Es gibt zwei vordefinierte Funktionen `Tx` und `Ty`, die bei Angabe von Offsets bei `setpoint` Kommandos dazu dienen, eine Koordinatentransformation nach folgender Formel durchzuführen:

`setpoint x[a,b]` $\longrightarrow$ `setpoint x[Tx(a,b),Ty(a,b)]`

Diese Funktionen sind als

```
def Tx(a,b) = a;
def Ty(a,b) = b;
```

vordefiniert. Im Normalfall wird also beim Erzeugen eines `setpoint`-Offsets durch die Funktion `Tx` der x-Koordinatenanteil ausgeblendet, durch die Funktion `Ty` der y-Koordinatenanteil. Durch Umdefinition dieser Abbildung lassen sich Effekte wie Transformationen, Skalierungen, Drehungen, etc. auf einfache Weise erzeugen. Konsultieren Sie für diese Möglichkeiten am Besten ein Buch zur grafischen Datenverarbeitung wie *Computer Graphics: Principles and Practice* von Foley/van Dam/Feiner/Hughes.

undef

Mit dem Schlüsselwort `undef` lassen sich mittels `def` vereinbarte Variablen und Funktionen wieder löschen.

setdefaults

Stellt die wichtigsten Parameter auf die Standardwerte zurück, wie in Abbildung 4.3 zu sehen. Außerdem werden alle Variablen- und Funktionsdefinitionen gelöscht!

Liniendicke	0.4 *pt*
Linienmuster	durchgezogen
Pfeilspitze	1.2 *pt* breit, 6.4 *pt* lang
Pfeilart	Geradenstücke, keine Ablenkung
Punktdicke	2.4 *pt*
Linienenden	abgerundet (round)
Punktzähler	0
`\unitlength`	1 *pt*

Abbildung 4.3: Standardwerte der Grafikparameter

setunitlength ⟨*dimen*⟩

Stellt den multiplikativen Faktor auf den Offset des `setpoint`-Befehls ein und bestimmt dadurch die Größe des Bildes. Beachten Sie bitte, daß per Voreinstellung durch `setdefaults` ein Wert von 1 *pt* eingetragen wird und deshalb auch relative Faktoren in der Form von Gleitpunktwerten jederzeit benutzt werden können. Bei Verwendung von expliziten Maßangaben wird dieser Faktor natürlich *nicht* aufmultipliziert!

setdotsize ⟨*dimen*⟩

Stellt den Punktdurchmesser auf ⟨*dimen*⟩.

setlinewidth ⟨*dimen*⟩

Stellt die Liniendicke auf ⟨*dimen*⟩.

setdash ⟨*dimen*⟩ ⟨*dimen*⟩ ...

Stellt das Linienmuster ein. Ohne Parameter wird das Muster zurückgesetzt auf durchgehend gezeichnete Linien. Ansonsten geben die Parameter abwechselnd die Strecken an, die schwarz bzw. weiß gezeichnet werden. Wird nur ein Parameter angegeben, ist das Muster gleichförmig schwarz-weiß, wie z.B. bei `setdash 2pt`.

defpattern ⟨*byte*⟩,⟨*byte*⟩,⟨*byte*⟩,⟨*byte*⟩,⟨*byte*⟩,⟨*byte*⟩,⟨*byte*⟩,⟨*byte*⟩

Definiert ein 8×8-Füllmuster für den `patclip`-Befehl. Die Parameter sind Dezimalzahlen im Bereich 0–255. Jedes gesetzte Bit in der Binärdarstellung der Zahl ergibt ein gesetztes Pixel im Füllmuster. Jede Zahl stellt eine Zeile des Füllmusters dar, wobei die erste Zahl der obersten Zeile entspricht. Es müssen zur Zeit genau acht Zahlen angegeben werden!

setlinecap ⟨*plain*|*round*|*arrow*|*wedge*|*triangle*⟩ ⟨*plain*|*round*|*arrow*|*wedge*|*triangle*⟩

Stellt das Aussehen von Linienenden ein. Der erste Parameter steht für den Linienanfang, der zweite für das Linienende. Die fünf möglichen Werte bedeuten „plain“ für eine abgeschnittene Linie, „round“ für eine Linie mit abgerundeten Enden, „arrow“ für eine Linie mit geschlossener ausgefüllter Pfeilspitze, „wedge“ für eine Linie mit offener Pfeilspitze ohne Abschluß und „triangle“ für eine Linie mit geschlossener ungefüllter Pfeilspitze.

setarrowsize ⟨*dimen*⟩ ⟨*dimen*⟩

Stellt die Größe von Pfeilspitzen ein. Der erste Parameter gibt an, wie weit sich die Schenkel des Pfeils vom Stamm entfernen, der zweite Parameter gibt die Länge der Pfeilspitze an.

setarrowshape ⟨*real*⟩ ⟨*real*⟩

Stellt das Aussehen von Pfeilspitzen ein. Die Schenkel und der Abschluß der Pfeilspitzen bestehen aus drei Bézierkurven mit doppeltem Kontrollpunkt auf halber Länge der Kurve. Die beiden Parameter beeinflussen die Lage dieser Kontrollpunkte, wobei der erste Parameter die Schenkel, der zweite den Abschluß steuert. Die Parameter sind Gleitpunktzahlen, wobei das Vorzeichen über die Richtung der Ablenkung der Bézierkurven entscheidet. Positive Werte ziehen die Schenkel und den Abschluß nach innen in Richtung Schwerpunkt des Dreiecks, das die Pfeilspitze umschließt. Negative Werte drücken entsprechend nach außen. Bei Aufruf ohne Parameter wird auf die Standardeinstellung zurückgesetzt (Pfeil aus Geradenstücken), bei Aufruf mit einem Parameter wird nur der Schenkelparameter gesetzt.

setarrowline ⟨*dimen*⟩

Bei Linienenden von Typ `triangle` steuert dieser Befehl die Liniendicke der Pfeilspitze.

setpoint ⟨*point number*⟩ ⟨*repeated*⟩ ⟨*position*⟩ ...

Setzt einen oder mehrere der 500–1000 internen Kontrollpunkte (je nach Gerät). Der Parameter *point number* steht dabei für die Nummer des zu definierenden Punktes.

Falls mit einem `setpoint`-Kommando mehrere Punkte definiert werden, können die definierenden Ausdrücke durch Leerzeichen oder, falls wegen Eindeutigkeit notwendig, durch `,` getrennt werden.

Der Parameter *repeated* ist ein Wiederholfaktor, der die Erzeugung mehrerer Punkte mit aufeinanderfolgenden Indizes bewirkt. Ein Wiederholfaktor hat folgenden rekursiven Aufbau:

```
<repeated>::=<> |
             <repeated> |
             <identifier>=<expr1>..<expr2>:<expr3>
```

Der Wiederholfaktor kann also optional (leer) oder verschachtelt angegeben werden. Ein einfacher Wiederholfaktor besteht aus einem noch nicht definierten Bezeichner und drei Ausdrücken, wobei der erste `expr1` und zweite `expr2` den Parameterbereich des Bezeichners definieren und der dritte die Anzahl zu erzeugender Punkte. Der Parameterbereich wird in äquidistante Abschnitte zerlegt und der Bezeichner erhält für jeden erzeugten Punkt einen dieser Werte im Parameterbereich. Durch Verwendung des Bezeichners im folgenden *position*-Parameter können die Punkte auf diese Weise an verschiedene Positionen gesetzt werden.

Optional kann in eckigen Klammern ein Offset zur aktuellen Position angegeben werden, der folgendes Aussehen hat:

```
<position>::=<> |
             [<expr1>,<expr2>] |
             [<expr1>,<expr2>,<rotate>]

<rotate>::=<expression> |
          <expression>[<ex1>,<ex2>]
```

In eckigen Klammern wird durch die Ausdrücke `expr1` und `expr2` ein Verschiebungsfaktor in x- und y-Richtung angegeben. Dieser Wert wird entweder mit dem mittels `setunitlength` eingestellten Wert multipliziert oder absolut durch eine bemaßte Größe angegeben und ergibt den tatsächlichen Offset. Falls ein *repeated* Parameter vorausgegangen ist, kann durch Angabe eines Rotationsfaktors der erzeugte Punkt noch gedreht werden. Der Bezugspunkt für die Drehung ist entweder der letzte eingegebene Punkt oder der durch `ex1` und `ex2` spezifizierte Punkt. Gedreht wird in beiden Fällen um den Winkel `expression`, der im Bogenmaß angegeben werden muß.

Für Beispiele lesen Sie bitte das Grafiktutorial, Anhang A.

⟨*pointlist*⟩

Für die nun folgenden Befehle wird als Parameter jeweils eine Punktliste erwartet. Die Punktliste bezieht sich auf die mit `setpoint` vergebenen Nummern, falls ein direkter Bezug auf diese Punkte gewünscht ist. Einzelne Punkte in der Punktliste können wegen Eindeutigkeit durch `,` getrennt werden. Bei Eindeutigkeit ist das Trennen durch Leerzeichen statthaft. Sollen mehrere direkt hintereinander definierte Punkte verwendet werden, reicht die Angabe des Bereichs, z.B. 105–113. Bereiche und Einzelpunkte können in der Punktliste beliebig gemischt werden.

Gibt man statt einer Punktnummer den Ausdruck $[p, q]\lambda$ an, so wird damit der Punkt $p(1 - \lambda) + q\lambda$ spezifiziert, also ein Punkt auf der Verbindungsgeraden von p und q. Für λ darf ein beliebiger Ausdruck aus den vier Grundrechenarten und runden Klammern eingesetzt werden. Selbstverständlich können die Punkte p und q wieder von der Form $[r, s]\mu$ sein.

Eine weitere Möglichkeit zur Angabe eines Punktes besteht in einem Ausdruck der Form (p, q). In diesem Fall hat der erzeugte Punkt die x-Koordinate von p und die y-Koordinate von q. Auch diese Notation darf wieder beliebig verschachtelt verwendet werden. Als Kurznotation ist statt der Angabe von p oder q das Zeichen `@` zulässig, das den zuletzt definierten Punkt der Punktliste einsetzt.

Durch Angabe von `@`, direkt gefolgt von einer Zahl ⟨n⟩, bezieht man sich auf die ⟨n⟩ zuletzt eingegebenen Punkte. Ohne Angabe einer Zahl bezieht sich `@` auf den in der Liste vorangehenden Punkt. Ist der Betrag von ⟨n⟩ negativ, wird außerdem der Punktzähler um den Betrag von ⟨n⟩ zurückgesetzt.

Als Besonderheit kann man eine Punktliste auch mit einigen speziellen Parametern versehen, die das Zeichnen von Teilobjekten und das wiederholte Zeichnen von Objekten erlauben. Ein Linienbefehl mit einer solchen modifizierten Punktliste hat dann folgendes Aussehen:

⟨`command`⟩ ⟨`pointlist`⟩ ⟨`pieces`⟩ ⟨`repeated`⟩ ⟨`position`⟩

Das Aussehen der Parameter `repeated` und `position` entspricht den gleichbezeichneten Parametern des `setpoint`-Kommandos. Allerdings ist die Bedeutung dahingehend geändert, daß das übergeordnete Kommando `command` entsprechend dem `repeat`-Faktor wiederholt wird, und zwar an den durch den `position`-Parameter bezeichneten Stellen. Ein `position`-Parameter darf nur angegeben werden, falls ein `repeated`-Parameter vorausgeht!

Der Parameter `pieces` kann verschachtelt angegeben werden und hat folgendes Aussehen:

```
<pieces>::=:<expr1>..<expr2> |
           :<expr1>..<expr2><pieces>
```

Durch die Parameter `expr1` und `expr2` wird nur ein Teil der Punktliste zum Zeichen des mit `command` erzeugten Objektes herangezogen. Die Werte

können sich auf die Punktnummern der Punkteliste beziehen und dürfen in diesem Fall bei n Punkten im Bereich $0 \leq$ `expr` $\leq n$ liegen. Bei bemaßten Werten wird das Zeichnen des Objekts bis zur Position `expr1` unterdrückt und dann nur bis Position `expr2` gezeichnet.

Für Beispiele lesen Sie bitte das Grafiktutorial, Anhang A.

dot ⟨*pointlist*⟩

Die Punkte der Punktliste werden mit dem voreingestellten Durchmesser als gefüllte Kreise dargestellt.

spline ⟨*pointlist*⟩

Zeichnet einen natürlichen Spline, der durch die Kontrollpunkte in ⟨*pointlist*⟩ definiert ist.

hermitespline ⟨*pointlist*⟩

In der Praxis hat man häufig das Problem, Kurven zeichnen zu müssen, die eine bestimmte Anfangs- und/oder Endrichtung haben sollen. Der Grafikbefehl zum Setzen eines solchen Hermiteschen Splines ist ist analog zum `spline`-Befehl, nur daß er eben `hermitespline` heißt. Bitte lesen Sie die Abschlußbemerkung zu Hermiteschen Splines!

lefthermitespline, righthermitespline ⟨*pointlist*⟩

Es ist auch möglich, nur an einem Ende des Splines die Richtung vorzugeben, und am anderen Ende des Splines eine natürliche Randbedingung zu fordern. Dies ist durch Verwendung von `lefthermitespline` und `righthermitespline` möglich.

! → Die Befehle zum Zeichnen Hermitescher Splines haben als Parameter eine Kontrollpunktliste, wobei der erste bzw. letzte bzw. erste und letzte Punkt der Liste eine Richtung vorgeben, die umso stärker ausgeprägt ist, je weiter der Punkt von der Kurve entfernt ist.

closedspline ⟨*pointlist*⟩

Durch diesen Befehl wird ein glatter geschlossener natürlicher Spline gezeichnet, wobei der erste und letzte Punkt der Punktliste miteinander verbunden werden.

poly, closedpoly ⟨*pointlist*⟩

Diese beiden Befehle zeichnen einen offenen bzw. geschlossenen Polygonzug, der durch die jeweilige Punktliste definiert ist.

defclip, clip, whiteclip, patclip, endclip

Diese Funktionen dienen zum Definieren eines Clippfades, bezüglich dessen Begrenzungslinie bis zum `endclip`-Kommando folgende TEX- und Grafik-Ausgaben gekappt werden. Die Begrenzungslinie *muß* geschlossen und in einem durchgehendem Stück definiert sein, unterliegt aber ansonsten keinerlei Beschränkungen. „Durchgehendes Stück“ soll in diesem Zusammenhang bedeuten, daß man nicht nachträglich auseinanderklaffende Stücke mit einem dritten schließt, sondern die Umrißlinie „wie mit einem Pinselstrich“ erzeugt. Durch Angabe von `defclip` wird der Beginn des Clippfades markiert. Alle folgenden Angaben definieren diesen Clippfad, bis ein `clip`-, `whiteclip`- oder `patclip`-Kommando die Eingabe des Clippfades abschließt. Alle folgenden TEX-Ausgaben, *also auch jeder Text*, werden bezüglich dieses Pfades gekappt, bis das `endclip`-Kommando diesen Modus beendet.

Beim Clippen beeinflussen die drei Clipbefehle die Ausgabe wie folgt:

`clip`	bisherige Elemente der Zeichnung werden nicht verdeckt
`whiteclip`	der Clipbereich wird gelöscht
`patclip`	der Clipbereich wird mit dem durch `defpattern` definierten Füllmuster ausgefüllt

input "*filename*"

Der Treiber liest die Grafikkommandos aus der Datei *filename*. Die Angabe des Dateinamens muß den Konventionen des verwendeten Betriebssystems entsprechen. Die Definition eines Pfades für solche externen Grafikdateien ist möglich. Lesen Sie dazu die entsprechenden Kapitel beim verwendeten Treiber!

! → Bitte beachten Sie, das das "-Zeichen bei manchen Macropaketen wie den Anpassungen an die deutsche Sprache `german.sty` eine Spezialfunktion bekommen hat, die die normale Verwendung wie beim `input`-Befehl unmöglich macht! In solchen Fällen muß vor Absetzen des `input` die Spezialbedeutung ausgeschaltet werden. Bei `german.sty` beschieht das z.B. durch eine Klammerung mit `\originalTeX` und `\germanTeX` Befehlen.

4.2 Macros zur Grafikeinbindung

Um Ihnen das Kopfzerbrechen zu ersparen, alle `\special`-Befehle zu lernen bzw. Ihre Grafiken auszumessen und mit `\hbox` und `\vbox` Befehlen zu hantieren, um Platz für Grafiken zu schaffen, gibt es eine ganze Anzahl vorgefertigter TEX-Macros, die Ihnen diese Arbeit abnehmen. Diese Macropakete sollten im Standard-Suchpfad von TEX liegen und können mit `\input` ⟨`name`⟩ geladen werden, wobei `name` für `graphic`, `bildmac`, `epsf`, etc. steht.

4.2.1 Die GRAPHIC-Macros

Die Datei `GRAPHIC.TEX` enthält die Macros zur Verwendung von Grafiken im Fremdformat.

IMG-Grafiken werden auf dem Atari ST zunächst mit IMGTOTEX.TTP bearbeitet. Dabei wird die gewünschte Auflösung der Grafik in dpi erfragt und in der Datei eingetragen und eine Zusatzdatei erstellt, die die Größe des Bildes in TEX-Einheiten enthält. Als Richtwert zur Auflösung kann bei Laserdruckern ein Wert von 150 dpi für Grafiken, die den ganzen Atari Bildschirm enthalten, angegeben werden. Die erzielte Größe im Ausdruck liegt in der Breite in etwa richtig für ein LATEX-Dokument. Als Beispiel siehe Abbildung B.1. In der TEX-Eingabe wird die bearbeitete Grafik dann mittels \includegraphic{filename} eingebunden. Der Platz für das Bild wird automatisch freigehalten. Zudem kann das Bild behandelt werden, wie ein einzelner Buchstabe, d.h. zentriert werden, etc.

An der Universität Augsburg wurde im Rahmen einer Diplomarbeit das objektorientierte Zeichenprogramm PCDRAW erstellt, das PostScript-Code als Ausgabe erzeugen kann. Zudem ist ein ähnlicher Mechanismus, wie für IMG-Grafiken auf dem Atari eingebaut. Mittels des Menüpunktes „Picture to TEX“ wird interaktiv die beabsichtigte Bemaßung abgefragt, der PostScript-Code des Bildes passend abgespeichert und eine Zusatzdatei für TEX erstellt. Das Bild kann dann mittels \psdraw{filename} in den Text eingebunden werden. Auch hier kann das Bild wie ein einzelner Buchstabe behandelt werden. Genaueres entnehmen Sie bitte der Beschreibung von PCDRAW.

Falls man selbst in PostScript programmiert, kann man sein fertiges Bild per Hand ausmessen und in eine zweite Datei die Größeninformation folgendermaßen ablegen:

```
\grwd = 108.160 mm
\grht = 67.600 mm
```

Dabei muß \grwd die Breite des Bildes und \grht die Höhe des Bildes zugewiesen bekommen. Wenn z.B. die erste Datei mit dem PostScript-Bild unter ESCHER.PS und die zweite Datei mit der Größeninformation unter ESCHER.TEX abgespeichert sind, so kann durch den Aufruf

```
\includeps{ESCHER}
```

dieses Bild in einen TEX-Text eingebunden werden, wobei automatisch genügend Platz freigehalten wird. Ein Beispiel für diese Technik ist Bild 4.4 von Peter Henderson und Peter Bumbulis frei nach Maurits C. Escher.

Bitte beachten Sie, daß dieser Mechanismus zum Einbinden von Bildern, die mittels PostScript erstellt wurden, inzwischen veraltet ist und nur aus Kompatibilitätsgründen unterstützt wird. Neue Projekte sollten gleich auf die EPSF-Macros zurückgreifen (siehe Abschnitt 4.2.3), die übrigens von Tomas Rokicki und Donald E. Knuth für DVIPS, einen anderen PostScript-DVI-Treiber, entwickelt wurden.

4.2.2 Die BILDMAC-Macros

Die Datei BILDMAC.TEX enthält einige Macros zur Verwendung der eingebauten Grafikbefehle. Die Entwicklung an diesen Macros dauert noch an. Die

Abbildung 4.4: Eine flächenfüllende Grafik nach Escher

Macros können jedoch als eine Anregung verstanden werden, eigene Macros zu definieren.

Trotz der noch andauernden Entwicklung empfehlen die Autoren dringend die Verwendung dieser Macros anstelle der ausführlichen `\special`-Befehle, da durch jeden `\special`-Befehl mit anschließenden Zeilentrenner ein Leerzeichen erzeugt wird, das das ordnungsgemäße Funktionieren des `\put`-Befehls beeinträchtigt.

Die Kompatibilität von neuen `BILDMAC`-Dateien zu vorherigen Versionen wird ab Treiberversion 2.2 weitestgehend bewahrt. Falls dies aus irgendeinem Grund einmal nicht möglich sein sollte, ist im Handbuch eine Möglichkeit angegeben, dieselben Ausgaben mit anderen Konstruktionen zu erreichen.

Das prinzipielle Schema der `BILDMAC`-Befehle ist immer gleich. Die jeden Grafikbefehl einleitende `\special{gr`-Sequenz und etwaiger Leerraum, der den Befehlen folgt, wird herausgefiltert. Auf diese Weise wird

```
\special{gr setpoint 0}
```

zu

```
\setpoint{0},
```

usw. Bitte drucken Sie sich die Datei `BILDMAC.TEX` aus und sehen selbst, welche Befehle zur Verfügung stehen. Macros, die über Textersatz hinausgehen, sind jeweils (hoffentlich) kommentiert.

4.2.3 Die EPSF-Macros

In den Dateien `EPSF.TEX` bzw. `EPSF.STY` liegen Macros vor, die zur Einbindung von Bildern in Encapsulated PostScript dienen. Wenn man eine PostScript-Datei vorliegen hat, sollte man zunächst feststellen, ob sie in Encapsulated PostScript-Form vorliegt. Falls dies nicht der Fall ist, muß man die Datei in diese Form bringen. Laden Sie also die Datei mit der Grafik in einen Editor. Sollte es dabei Probleme mit der Zeilenlänge geben, hat das erzeugende Programm den PostScript-Standardzeilentrenner LF verwendet und Sie müssen die Datei mit `UNIX2DOS` oder `CRLF`[1] bearbeiten, so daß eine Standard-ASCII-Datei entsteht, die Ihr Drucker übrigens problemlos verdaut. Haben Sie die Datei erfolgreich geladen, sollte unter den ersten paar Dutzend Zeilen ein PostScript-Kommentar

```
%%BoundingBox: 1 2 3 4
```

oder zumindest

```
%%BoundingBox: (atend)
```

zu finden sein. Im zweiten Fall kopieren Sie den richtigen `Bounding Box` Kommentar vom Dateiende an den Dateianfang und löschen die jetzt obsolete `atend`-Zeile. Die Datei ist jetzt problemlos durch TEX und `DVILW` zu verarbeiten.

Haben Sie keinen `%%Bounding Box`-Kommentar gefunden, müssen Sie selbst einen einfügen. Dazu steht Ihnen die Hilfsdatei `BB.PS` von Bernie Cosell zur Verfügung. Kopieren Sie den Inhalt von `BB.PS` und Ihres Bildes in eine neue Datei und drucken dieses Bild aus. Dazu müssen Sie evtl. am Ende der Datei ein `showpage`-Kommando einfügen. Das neue Bild hat jetzt einen Rahmen und die notwendigen Parameter für die `Bounding Box` stehen am linken unteren Ende der Grafik. Laden Sie nun erneut das ursprüngliche Bild und fügen irgendwo am Anfang der Datei eine Zeile

```
%%BoundingBox: 1 2 3 4
```

ein, wobei Sie anstelle von `1`, `2`, `3` und `4` die richtigen Werte von Ihrem Testausdruck übernehmen. Falls Ihre Datei eine `showpage`-Anweisung enthielt, brauchen Sie diese nicht entfernen, da das Ausdrucken der Grafik jetzt im Zusammenhang mit dem TEX-Text geschieht und die entsprechenden Befehle von `DVILW` erzeugt werden, d.h. das `showpage` wird lokal von `DVILW` umdefiniert.

Zum Setzen des Bildes im laufenden Text benötigt TEX einige Macros, die — wie schon vorher erwähnt — in der Datei `EPSF.STY` gespeichert sind. Laden Sie diese Datei durch

```
\input epsf.sty
```

(TEX) oder durch

[1] Quellcode wird mitgeliefert

```
\documentstyle[...,epsf,...]{...}
```

(LaTeX). Sie können natürlich in TeX auch die Datei `EPSF.TEX` werwenden und sich die Endung beim `\input`-Befehl sparen. Der Inhalt der Dateien ist gleich! Jetzt können Sie Ihr Bild durch

```
\epsfbox{bild.ps}
```

setzen lassen, wobei durch TeX bzw. LaTeX automatisch durch die `Bounding Box`-Anweisung genügend freier Platz gelassen wird. Außerdem haben Sie die Möglichkeit, das PostScript-Bild in der Größe dem Text anzupassen. Dazu müssen Sie lediglich *vor* dem `\epsfbox`-Kommando[2] eines der Kommandos

```
\epsfxsize=<dimen>
\epsfysize=<dimen>
```

absetzen, um das Bild horizontal bzw. vertikal zu skalieren. Der Parameter `<dimen>` ist eine gültige TeX-Maßeinheit wie 150pt oder 5.7cm. Falls Sie beide Kommandos absetzen, hat das `\epsfxsize`-Kommando Vorrang. Dadurch können keine Verzerrungen entstehen! Ein günstiger Wert für die horizontale Skalierung ist z.B. für TeX `\hsize` bzw. für LaTeX `\textwidth`. Dadurch ist das Bild genauso breit, wie der umgebende Text.

! → Bei Verwendung von LaTeX kann es bei manchen Environments notwendig sein, unmittelbar vor das `\epsfbox`-Kommando ein `\leavevmode`-Kommando zu stellen. Sie werden die entsprechende Fehlermeldung bekommen und können ganz gelassen reagieren!

Durch Absetzen des Befehls `\epsfverbosetrue` *vor* den `\epsfbox`-Befehlen können Sie sich die Größe der eingebundenen PostScript-Grafiken am Bildschirm anzeigen lassen.

[2] Achtung: in älteren Versionen der `EPSF`-Macros heißt `\epsfbox` `\epsffile`.

Anhang A. Grafiktutorial

Grundlagen: Punkte, Linien, Einheiten und Macropakete

DVI und DVILW erzeugen ihre Grafiken dadurch, daß man einige Punkte durch ihre Positionen definiert und diese dann durch verschiedene Linien miteinander verbindet oder durch verschieden dicke Punkte sichtbar macht. Dazu folgendes Plain-TEX-Beispiel:

```
\leftline{\hskip 3cm\special{gr setpoint 0}}
\rightline{\special{gr setpoint 1; poly 0,1}
  \hskip 3cm}
```

Dadurch entsteht eine ähnliche Abbildung wie links.

Verwendet wurden in diesem Beispiel zwei \special-Befehle des Treibers, nämlich setpoint, womit die Positionen der Punkte 0 und 1 festgelegt wurden und poly, der die Verbindung der beiden Punkte 0 und 1 durch eine Gerade zeichnete. Um die Positionierung etwas zu erleichtern, werden folgende TEX-Macros und Variablen für Plain-TEX definiert:

```
\newdimen\unitlength \unitlength = 1pt
\def\setunitlength#1{%
  \unitlength=#1\special{gr setunitlength #1}}

\def\picture(#1,#2)#3{\vbox to #2\unitlength{\vss\hbox
  to #1\unitlength{#3\hss}}}

\def\put(#1,#2)#3{\unskip\raise#2\unitlength\hbox
  to 0pt{\kern#1\unitlength#3\hss}\ignorespaces}
```

Für LATEX ist natürlich die Definition der Macros \picture und \put unnötig, da das picture-Environment genau dasselbe leistet. Der nächste Absatz muß deshalb von LATEX-Anwendern übersetzt werden, d.h. mit den notwendigen \begin{} und \end{}-Zusätzen versehen werden.

Die neue Variable \unitlength gibt die Maßeinheit an, in der alle Punktekoordinaten angegeben werden können. Das Macro \setunitlength dient der Veränderung dieser Maßeinheit, zusätzlich wird den Treibern diese Veränderung über das Grafik-Kommando setunitlength mitgeteilt. Es ist wichtig, sich darüber klarzuwerden, daß die TEX-Variable \unitlength für die Treiber unbekannt ist, und nur für die beiden Macros \picture und \put von Bedeutung ist. Alternativ ist auch die Verwendung von absoluten Längenangaben, wie sie TEX versteht (z.B. 1.1 *cm*, 5 *pt*), erlaubt.

Mit \picture(w,h) wird Platz für ein Bild der Breite w und der Höhe h geschaffen. Mit \put(x,y) können Bildelemente innerhalb von \picture positioniert werden. Beispiel: Die Befehle

```
\setunitlength{1cm}
\centerline{\picture(3,2){\put(0,0){a}\put(3,2){b}}}
```

b erzeugen das Bild links.

Anstatt der Buchstaben a und b sollen nun die Endpunkte einer Geraden an dieselben Stellen gesetzt werden. Dies geschieht mit folgenden Befehlen, die das Bild links liefern:

a

```
\setunitlength{1cm}
\centerline{\picture(3,2){
  \put(0,0){\special{gr setpoint 0}}
  \put(3,2){\special{gr setpoint 1; poly 0,1}}}}
```

Punktdefinitionen für fortgeschrittene Aufgaben

Es ist jedoch nicht nötig, jede einzelne Punkteposition durch einen neuen `\special`-Befehl festzulegen, vielmehr ist das Kommando `setpoint` selbst in der Lage, Punktpositionen durch Angabe von Koordinaten zu definieren. Das obige Bild hätte auch so erzeugt werden können:

```
\setunitlength{1cm}
\centerline{\picture(5,2){\special{gr setpoint 0 1[4,2];
  poly 0,1}}}
```

Dieses Beispiel soll zeigen, daß es möglich ist, beliebig viele Punktpositionen durch ein einziges `setpoint`-Kommando festzulegen. Zum anderen sieht man, daß es möglich ist, zur Punktenummer zusätzlich eine x- und eine y-Koordinate anzugeben. Die Maßeinheit dieser Koordinatenangaben wird durch das Kommando `setunitlength` festgelegt, falls die Angaben Gleitpunktwerte sind. Wahlweise könnte auch eine andere gültige TEX-Längenangabe wie `0.3 true cm` benutzt werden. In diesem Fall wurden die Offsets durch das TEX-Macro `\setunitlength` festgelegt.

Eine weitere Möglichkeit, das obige Bild zu erzeugen, ist:

```
\setunitlength{1cm}
\centerline{\picture(5,2){\special{gr setpoint 0, [4,2];
  poly 0,1}}}
```

Wie man sieht, ist die Punktenummer des zweiten gesetzten Punktes weggelassen, allerdings steht nun ein Komma zwischen beiden Angaben, da dies sonst wie `0[4,2]` gelesen würde. Werden Punktenummern nicht angegeben, so numerieren `DVI` und `DVILW` die Punkte automatisch in aufsteigender Reihenfolge. Dies ermöglicht auch folgende Eingabe:

```
\setunitlength{1cm}
\centerline{\picture(5,2){
  \special{gr setpoint [0,0] [4,2]; poly @2}}}
```

Hier sind überhaupt keine Punktenummern mehr angegeben. Natürlich ist diesem Zusammenhang nicht mehr unbedingt klar, welche Punktenummern der Treiber verwendet hat, denn dies hängt ja davon ab, welche Punktenummern in vorangegangenen Bildern verwendet wurden. Deshalb wurde im `poly`-Kommando anstatt einer Liste von Punkten der Ausdruck `@2` verwendet, der besagt, daß die Punkteliste aus den beiden zuletzt gesetzten Punkten besteht. Auch der Ausdruck `@-2` wäre an dieser Stelle möglich gewesen. In diesem Fall wäre der interne Punktezähler des Treibers wieder um den Wert 2 zurückgesetzt worden.

Die nächste Aufgabe besteht darin, Punkte auf einer Linie als Ausgangspunkte anderer Linien zu verwenden, wie links im Bild zu sehen ist.

Der Code, der dieses Bild erzeugte, hat folgendes Aussehen:

```
\setunitlength{1cm}
\centerline{\picture(3,3){
  \special{gr setpoint 0, [3,3], [0,3];
    poly 0,1; poly [0,1]1/2, 2;}}}
```

Setzt man statt einer Punktnummer den Ausdruck $[p,q]\lambda$ ein, so wird damit der Punkt $p(1-\lambda)+q\lambda$ spezifiziert. `[0,1]1/2` ist also der Punkt genau in der Mitte zwischen den Punkten 0 und 1. Für λ kann übrigens ein beliebiger Ausdruck[1] stehen, der z.B. aus reellen Zahlen, den vier Grundrechenarten und runden Klammern gebildet wird. Anstatt der Punktenummern p und q kann auch wieder ein Ausdruck der Form $[p',q']\lambda'$ stehen.

Jetzt soll folgendes Problem gelöst werden: Zwei Quadrate sind wie in der Abbildung links gegeben, die übrigens folgendermaßen erzeugt wurde:

```
\setunitlength{1cm}
\centerline{\picture(3,3){
  \special{gr setpoint 0[0,2] [1,2] [1,3] [0,3]
                         [2,0] [3,0] [3,1] [2,1];
              closedpoly 0-3; closedpoly 4-7}}}
```

Das linke obere Quadrat hat die Eckpunkte 0,1,2,3 das rechte untere die Eckpunkte 4,5,6,7. Nun soll ein Pfeil von der Mitte der Unterkante des linken Quadrats zur Mitte der linken Kante des rechten Quadrats gezeichnet werden. Dies geschieht durch folgenden zusätzlichen Befehl im `\special`-Kommando mit dem Ergebnis, das links zu sehen ist.

```
setlinecap round arrow;
poly [0,1]1/2, ([0,1]1/2,[4,7]1/2), [4,7]1/2
```

Die Schreibweise (p,q) spezifiziert einen Punkt, der die x-Koordinate von p und die y-Koordinate von q besitzt. Selbstverständlich kann auch diese Schreibweise wieder beliebig geschachtelt werden. Das Kommando `setlinecap round arrow` dient lediglich der Darstellung von Pfeilspitzen am hinteren Ende einer Linie. Die verschiedenen Möglichkeiten, Linienenden zu gestalten, werden später gesondert besprochen. Eine andere Möglichkeit, den Pfeil zu erhalten, stellt folgendes `DVI`/`DVILW`-Kommando dar:

[1] Lesen Sie dazu auch den Abschnitt über Ausdrücke

```
poly [0,1]1/2, (@,[4,7]1/2), (4,@)
```

Das Symbol `@` steht bei einem Linienkommando wie `poly` für den in der Punkteliste zuletzt spezifizierten Punkt, im ersten Fall also für den Punkt `[0,1]1/2` und im zweiten Fall für den Punkt `([0,1]1/2,[4,7]1/2)`.

Das Zeichnen von Punkten

Bisher wurden die verwendeten Punkte als mathematisches Objekt ohne räumliche Ausdehnung betrachtet und durch Linien verbunden. Es ist jedoch auch möglich, an Punktepositionen gefüllte Kreise mit beliebigen Durchmesser zu zeichnen. Hierzu werden die Befehle `dot` und `setdotsize` verwendet. Wenn dieselben Punkte, die schon im Bild mit den Quadraten verwendet wurden, als Grafikpunkte der Dicke 2.4 *pt* dargestellt werden, ergibt sich das Bild links. Im Code wurden lediglich die beiden `closedpoly`-Anweisungen durch `dot 0-7` ersetzt.

Das Zeichnen von Linien

Die nun folgenden Beispiele sollen zeigen, welche Möglichkeiten bestehen, verschiedene Kurven durch definierte Punkte zu ziehen. Zunächst seien die folgenden 5 Punkte definiert:

```
setpoint 0[1,0] [2,0] [2,1] [0,1] [0,0]
```

Dann ergeben sich die Möglichkeiten von Abbildung A.1.

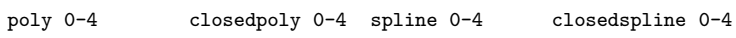

Abbildung A.1: Kurven durch Punkte

Spline-Kurven

Zunächst betrachten wir die dritte Kurve näher, den sog. **Spline**. Tja, und was ist nun ein Spline? Grob gesagt, eine Kurve, die irgendwie durch eine Anzahl von Kontrollpunkten, die die Kurve definieren, verläuft. Abbildung A.2 zeigt einen Spline, der durch zwei Endpunkte gegeben ist.

Solche Splines werden, wenn sie nicht nur durch zwei Punkte bestimmt sind, mit Hilfe von Bézierkurven gezeichnet. Eine solche Bézierkurve ist durch vier Punkte bestimmt, wie in Abbildung A.3 zu sehen ist. Die Bézierkurve läuft durch die beiden Endpunkte, jeweils in Richtung der beiden Kontrollpunkte, die in Abhängigkeit von ihrer Entfernung zur Kurve globale Kontrolle unterschiedlicher Stärke ausüben. Man kann sich die Kontrollpunkte als fest verankerte Magneten vorstellen, die die bewegliche Kurve anziehen, bis sich ein Gleichgewicht eingestellt hat.

Abbildung A.2: Spline mit zwei Endpunkten

Abbildung A.3: Bézierkurve

Soll ein Spline nun durch eine beliebige Anzahl von Punkten laufen, wird er aus vielen Bézierkurven zusammengesetzt. Die Lage der Kontrollpunkte wird dabei so bestimmt, daß die Übergänge immer glatt aussehen. Lediglich der erste Kontrollpunkt der ersten Bézierkurve und der zweite Kontrollpunkt der letzten Bézierkurve sind dadurch nicht bestimmt. Bei einem *natürlichen Spline* werden diese Kontrollpunkte so gewählt, daß der Spline in den Endpunkten keine Krümmung mehr hat. Abbildung A.4 zeigt einen natürlichen Spline durch drei Punkte. Die Kontrollpunkte der Bézierkurven sind hier nicht mehr zu sehen, diese werden ohnehin automatisch berechnet.

Abbildung A.4: Natürlicher Spline

Splines für fortgeschrittene Aufgaben

Die nächsten Beispiele sollen zeigen, daß es auch möglich ist, Splines mit vorgegebenen Ableitungen an den Enden zu zeichnen. Dazu seien die folgenden Punkte gegeben:

```
setpoint 0[0,0] [1,1] [2,2] [1,0] [1,2]
```

Die Punkte 3 und 4 sollen in diesem Beispiel die Ableitungen definieren. In der Praxis sieht dies aus, wie Abbildung A.5.

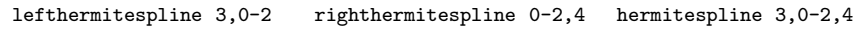

Abbildung A.5: Hermitesche Splines

Die Praxisvorgabe ist die Aufgabe, Kurven zeichnen zu müssen, die eine bestimmte Anfangs- und/oder Endrichtung haben sollen. Nehmen wir zum Beispiel an, der Spline in Abbildung A.4 soll im linken Endpunkt senkrecht nach oben hinauslaufen und im rechten Endpunkt senkrecht nach unten einmünden. Dies kann durch die zusätzliche Vorgabe der beiden noch freien Kontrollpunkte am Anfang und Ende geschehen. Ein solchen Spline wollen wir hier *Hermiteschen Spline* nennen. Abbildung A.6 zeigt das Ergebnis.

Abbildung A.6: Hermitescher Spline

Bei einem `lefthermitespline` dient der erste Punkt in der Liste dazu, die Richtung der Kurve zu definieren, d.h. die Kurve startet im *zweiten* Punkt der Liste in Richtung des *ersten* Punktes. Je weiter dieser erste Punkt vom Startpunkt entfernt ist desto ausgeprägter ist die Richtungsvorgabe. Ein `righthermitespline` ist das Analogon zum `lefthermitespline`, nur ist hier die Richtung im letzten Punkt der Liste vorgegeben. Die Kurve endet im *vorletzten* Punkt in die Richtung des *letzten* Punktes zeigend. Zwei solcher Kurven sind in Abbildung A.7 zu sehen. Die untere der beiden Kurven hat im linken Punkt eine vorgegebene Richtung, die obere Kurve im rechten Randpunkt.

Beim `hermitespline` sind die Richtungen in beiden Endpunkten der Kurve, also im *zweiten* und *vorletzten* Punkt der Liste vorgegeben. Die Kurve startet in Richtung des *ersten* angegebenen Punktes und endet in die Richtung des *letzten* angegebenen Punktes zeigend.

Eine weitere Sorte von Splines sind geschlossene Splines, wie einer in Abbildung A.8 zu sehen ist. Der Grafikbefehl ist — wie nicht anders zu erwarten

Abbildung A.7: Halbe Hermitesche Splines

— analog zum `spline`-Befehl `closedspline` und hat als Parameter eine Kontrollpunktliste. Der erste und letzte Punkt werden so miteinander verbunden, daß eine glatte Kurve entsteht.

Abbildung A.8: Geschlossener Spline

Das Zeichnen von Geraden

Bei all diesen bisher gezeichneten Splines handelt es sich um sogenannte *kubische* Splines. Es gibt allerdings auch noch die Möglichkeit *lineare* Splines zu zeichnen. Diese linearen Splines sind Polygone. Abbildung A.9 zeigt einen einfachen Polygonzug, Abbildung A.10 einen geschlossenen Polygonzug.

Abbildung A.9: Polygonzug

Die Befehle zum Zeichnen dieser Polygonzüge sind `poly` und `closedpoly` und erwarten als Parameter jeweils eine Punktliste.