Für unix/mail

Dr. Peter Vollenweider, Universität Zürich, CH-8057 Zürich

(1. Juli 1993)

PostScript -- 1. Teil: Die PostScript-Sprache

Ein deutsches PC-Magazin hat PostScript einmal als Chamäleon bezeichnet. Diese Bezeichnung ist gar nicht so schlecht, verfügt doch PostScript über sehr viele Facetten.

Zuerst einmal ist PostScript eine Seitenbeschreibungssprache, welche einzelne Druckseiten -- etwa im DIN-Format A4 -- beschreibt. Vereinfacht gesagt steuert PostScript einen Seitendrucker. Aber PostScript ist auch eine ausgewachsene Programmiersprache mit allen Attributen einer herkömmlichen Programmiersprache: z.B. Datentypen (einfache Zählvariablen, Arrays = Zahlenvektoren, Strings = Zeichenketten) und Kontrollmöglichkeiten (Bedingungen, Schleifen, Prozeduren).

Dann kann PostScript auch einzelne Illustrationen beschreiben; viele Software-Komponenten verwenden EPS-Files (Encapsulated PostScript Format) als Austauschformat. In der grafischen Industrie steht ein anderer Aspekt von PostScript im Vordergrund: die zahlreichen professionellen Fonts oder Schriftschnitte, die intern PostScript-mässig kodiert sind (Umrisse, B#ezierkurven, Adobe Font Technologie, Hints). Im Zusammenhang mit Workstations spricht man vom Imaging-Modell, welches angibt, auf welche Art und Weise grafische Objekte oder Elemente definiert und gezeichnet werden. Und schliesslich gibt es auch Display PostScript (DPS/X und NextStep), das für eine optimale Uebereinstimmung zwischen Druckausgabe und Bildschirmdarstellung sorgt.

Wie können Sie auf der Workstation PostScript-Code verarbeiten?

Für den Anfänger gibt es zwei Möglichkeiten, auf einer Workstation "PostScript" auszutesten und zu verarbeiten:

1. Ein Previewer dient dazu, entweder eine EPS-Datei oder aber PostScript-Druckseiten auf dem Bildschirm anzuzeigen. Eine EPS-Datei beschreibt eine einzelne hochwertige Grafik oder Illustration. "%%BoundingBox" ist der wichtigste Kommentar in einer Illustration. Er gibt das Rechteck an, in dem sich sämtliche gedruckten Elemente der Abbildung befinden. Verschiedene Previewer erzeugen eine Fehlermeldung, wenn das File keinen gültigen BoundingBox-Kommentar enthält. Allfällige Fehlermeldungen gehen ins Konsolenfile. Mehrseitige Dokumente tragen meistens die Kennung ".ps".
   NextStep bietet die beiden Previewer Yap und Preview (mit Adobe-Interpreter) an.
   Sun NeWS: der Previewer pageview setzt einen NeWS-Server voraus.

SGI: Silicon Graphics bietet den Previewer xpsview (mit Adobe-Interpreter) an. Die Nutzung von xpsview setzt einen DPS/X-Server voraus. DPS/X ist eine Kombination von X11 und PostScript und in den letzten zwölf Monaten so etwas wie ein Standard geworden (implementiert von IBM, Silicon Graphics, DEC, Sunsoft 1993).
GNU: Die beiden X-Clients gs (ghostscript) und ghostview setzen für die Anzeige lediglich einen X11-Server voraus. Das Besondere an ghostview ist, dass man/frau einzelne Seiten markieren und beliebig durchblättern kann -- falls es sich um ein mehrseitiges Dokument handelt. Siehe man-pages.

2. Interaktive Shell-Werkzeuge, die direkt mit dem PostScript-Interpreter kommunizieren. Diese Dienstprogramme schicken Befehl um Befehl an den PostScript-Interpreter und zeigen die Meldungen des Interpreters sofort an. Der (übliche) Prompt "PS>" deutet an, dass der Interpreter auf einen weiteren Input wartet. Wenn Sie einen unbekannten Befehl "dada" eintasten, meint der PostScript-Interpreter:
   %%[Error: undefined; OffendingCommand: dada]%%
   Den Dialog mit dem PostScript-Interpreter können Sie abbrechen, indem Sie ctrl-d eingeben.
   Nextstep bietet das Programm pft (mit Adobe-Interpreter) an.
   Sun NeWS: psh
   RS/6000: dpsexec (mit Adobe-Interpreter).

Wenn man ein PostScript-File auf einen Seitendrucker schicken, d.h. zuladen will, muss das PostScript-File am Schluss den Operator showpage enthalten. Fügen Sie diesen Befehl notfalls an! Dieser sorgt dafür, dass das Papierblatt schliesslich ausgespuckt wird. Und noch ein Tip: verzichten Sie darauf, nichtdruckbare Steuerzeichen wie z.B. ctrl-d direkt in Ihr PS-File hinein zu schreiben.

Denken Sie ferner daran, dass die aufgerufenen PostScript-Fonts auf Ihrer Workstation installiert sein müssen (ausser diese würden zusammen mit Ihrem PostScript-Programm zugeladen). Die Erfahrung zeigt, dass fehlende Fonts zum Stolperstein werden können; eine bekannte Fehlermeldung lautet "Font not found, using Courier", zu Deutsch: es lebe die Schreibmaschine.

Die Gliederung eines PostScript-Programms

Ein Schriftzug kann in unzähligen Varianten wiedergegeben werden. Das nachfolgende Programm *effekt.eps* demonstriert einen Texteffekt, der die Schriftzeichen von HELLO mit dem Muster "World-World-World" füllt. Sie brauchen dieses Programm jetzt noch nicht zu verstehen! Betrachten Sie nur dessen Gliederung und nehmen Sie zur Kenntnis, wie kompakt ein Texteffekt programmiert werden kann. Dieser Texteffekt beruht auf den Zeichenumrissen des Wortes HELLO in der Schriftart Helvetica-Bold.

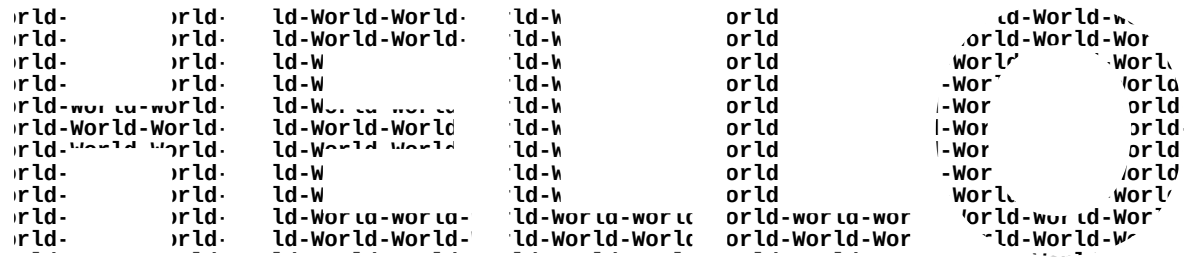

Abbildung 1: Texteffekt “HELLO World”

```
%!PS-Adobe-2.0 EPSF-2.0
%%DocumentFonts: Courier-Bold Helvetica-Bold
%%BoundingBox: 5 -5 540 120
%%Title: effekt.eps
%%EndComments

 /Normal /Courier-Bold findfont 10 scalefont def
 /Bold /Helvetica-Bold findfont 160 scalefont def

 /strg (World-World-World-World-World-World-World-World-World-World-
 World-World-World-World-World-World-World-World-World-World-World)
 def     % Zeichenkette

 /crlf   % carriage return, line feed
 { currentpoint 10 sub
   exch pop 0 exch moveto } def

 /prtstring { strg show crlf } def

 /Background
 { 16 { prtstring } repeat}  def     % Fuellmuster

%%EndProlog

 gsave
   newpath 0 0 moveto
   8 setflat
   Bold setfont (HELLO) true charpath clip
                        % bildet die Umrisse von HELLO und schneidet das Muster aus

   0 133 moveto
   Normal setfont Background      % Fuellmuster
 grestore

 showpage
```

Das Programm *effekt.eps* zeigt Ihnen die typische Struktur eines PostScript-Programms:

1. Der EPS-Kopf enthält einige Kommentare -- gekennzeichnet durch "%%". Der Kommentar "%%EndComments" schliesst den EPS-Kopf ab.
2. Der Prolog definiert die wichtigsten Funktionen als Prozeduren, z.B. die Prozedur *Background*, die ihrerseits die Prozedur prtstring sechzehnmal aufruft. Der Befehl show ist ein Text-Operator, der einen Schriftzug ausdruckt. Der Operator repeat führt den PostScript-Code zwischen den geschweiften Klammern entsprechend oft aus. Die Prozedur Background macht nichts anderes als "World-World-World" auf den Hintergrund zu schreiben. Der Prolog wird durch den Kommentar "%%EndProlog" abgeschlossen.
3. Das sogenannte Script führt nun eine Seitenbeschreibung durch, indem die vordefinierten Prozeduren sowie gewöhnliche PostScript-Operatoren verwendet werden. Der Operator *clip* sorgt dafür, dass die Zeichenumrisse mit dem Hintergrundmuster gefüllt werden.

Zum im Programm verwendeten Zeichensatz sind folgende Punkte anzumerken:

Mit dem %-Zeichen wird ein PostScript-Kommentar eingeleitet. Alle Zeichen bis zum Zeilenende werden dann vom PostScript-Interpreter als Kommentar angesehen und nicht mehr beachtet. Beispiel:

```
... % carriage return, line feed
```

Es gibt tatsächlich Grafik-Programme, die Code-Zeilen erzeugen, die nicht mit Newline (\n) abgeschlossen sind. Dies führt dazu, dass das gesamte PostScript-Programm als ein riesiger Kommentar interpretiert wird!

Zeichenketten stehen zwischen runden Klammern:

```
(World-World-World-)
```

Prozeduren werden zwischen geschweiften Klammern angegeben:

```
{ currentpoint 10 sub
  exch pop 0 exch moveto }
```

Die geschweiften Klammern weisen immer auf etwas Ausführbares hin. Der Schrägstrich bewirkt, dass die Prozedur (noch) nicht ausgeführt, sondern in den Stack gestellt wird. Die ganze Programmzeile definiert eine Prozedur namens *crlf:*

```
/crlf   % carriage return, line feed
{ currentpoint 10 sub
  exch pop 0 exch moveto } def
```

Die Prozedur crlf wird im Kapitel *Operanden-Stack* erläutert.

Nun möchte der 1. Teil die wichtigsten Konzepte der PostScript-Sprache einführen:

- Koordinatensystem
- Pfad-Konstruktion
- Operanden-Stack
- Der grafische Status
- Dictionary

Koordinatensystem

Der Zweck eines PostScript-Programms ist die Ausgabe von Text und Grafik auf einen Drucker oder einen Bildschirm. Der Nullpunkt des PostScript-Koordinatensystems befindet sich auf dem Papierblatt oder dem Bildschirm-Fenster links unten. Die x- und y-Achsen werden in typografischen Punkten angegeben (1/72 Zoll, Einfluss der grafischen Industrie!).

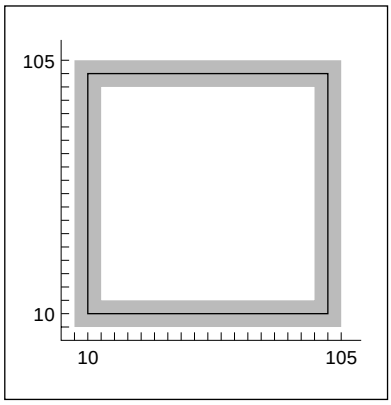

Abbildung 2: Koordinatensystem mit Nullpunkt unten links

Abb. 2 zeichnet ein Quadrat, dessen Strichstärke 10 Punkt beträgt.

Der Operator *translate* verschiebt den Nullpunkt des Koordinatensystems. Mit *scale* werden die x- und y-Achsen des Koordinatensystems neu skaliert. Der Operator *rotate* dreht das gesamte Koordinatensystem, Beispiel:

```
-6 rotate
```

Dieser Befehl dreht das aktuelle Koordinatensystem um sechs Grad. Das nächste Beispiel

```
.5 .5 scale
```

verkleinert das Koordinatensystem; beide Achsen werden proportional auf 50% verkleinert.

Das zweidimensionale Koordinatensystem bildet gewissermassen Ihr Spielfeld für die Bewegungen und Pfade, die Sie aufbauen möchten.

Pfad-Konstruktion

Wichtige Befehle der Pfadkonstruktion sind die Operatoren, die eine Linie, einen

Kreis oder eine B#ezier-Kurve beschreiben (lineto, rlineto, arc, curveto, rcurveto), die einen Pfad schliessen (closepath) oder die lediglich eine Bewegung ausführen (moveto). Mit *moveto-lineto* haben Sie noch keinen Strich ausgezogen, sondern erst einen sogenannten Pfad gebildet. Bevor Sie einen Strich ziehen können, müssen Sie einen Pfad bilden. Man spricht in diesem Zusammenhang auch vom Imaging-Modell. Ein Pfad ist gewöhnlich aus mehreren Segmenten zusammen-gesetzt.

Betrachten Sie das folgende Beispiel:

```
1 setlinewidth % Strichstaerke
0 0 100 0 360 arc % Pfad
stroke % ausziehen
```

Die Operanden werden links vom Operator angegeben. Das Prozentzeichen (%) leitet einen Kommentar ein, der sich jeweils bis zum Zeilenende (\n) erstreckt. Die Operatoren *arc* und *stroke* konstruieren einen Kreis mit einem Radius von 100 Punkt und ziehen diesen aus. Sein Zentrum befindet sich an der Position (0,0), also am Ursprung des Koordinatensystems. Die beiden Zahlen 0 und 360 geben Anfangs- und Endwinkel in Grad an; statt eines Kreises könnte also auch ein einzelnes Kuchenstück gezeichnet werden. Beispiel Bogenkurve:

```
0 0 100 0 90 arc % Bogen 90 Grad
stroke
```

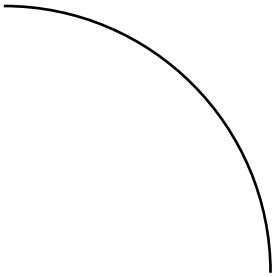

Abbildung 3: Bogenkurve von 0 bis 90 Grad

Wichtige Operatoren der Pfadkonstruktion im Ueberblick:

moveto	Bewegung zu einem Punkt hin
rmoveto	Bewegung, mit relativen Koordinaten
lineto	Linie zu einem Punkt hin
rlineto	Linie, mit relativen Koordinaten
arc	Bogenkurve, Kreis
arcn	Bogenkurve, Kreis, im Uhrzeigersinn
curveto	B#ezier-Kurve
rcurveto	B#ezier-Kurve, mit relativen Koordinaten
closepath	Pfad schliessen

"10 10 moveto” macht eine Bewegung zum Punkt (10,10) im Koordinatensystem. Ein Zahlenpaar wie "10 10” besteht immer aus einem X-Wert, der die horizontale

Position im Koordinatensystem angibt, und einem Y-Wert, der die vertikale Position angibt. Bei rmoveto, rlineto und rcurveto werden die Pfade nicht in absoluten Koordinaten (ausgehend vom Ursprung), sondern in relativen Koordinaten angegeben (ausgehend von der aktuellen Position). Die B#ezier-Kurve wird später vorgestellt werden. Der Operator curveto ist ausserordentlich wichtig, da die meisten Formen und Schriftzeichen aus B#ezier-Segmenten zusammengesetzt sind.

Operanden-Stack

PostScript ist eine Stack-orientierte Programmiersprache. Beim Operanden-Stack handelt es sich um einen Speicherbereich, der einem Programm zur Verfügung steht, um Objekte zwischenzeitlich abzulegen. Die PostScript-Operatoren (d.h. Befehle) nehmen ihre Operanden vom Stack und stellen allfällige Resultate ebenfalls in den Stack. Es gilt das LIFO-Prinzip (Last In, First Out). Ein PostScript-Programmierer muss gewissermassen vor seinem inneren Auge Rechenschaft ablegen über das Kommen und Gehen der Objekte auf dem Stack.

Wichtige Operatoren, die den Operanden-Stack manipulieren:

dup	oberstes Objekt duplizieren
exch	zwei oberste Objekte vertauschen
pop	oberstes Objekt vernichten
copy	n Objekte duplizieren
index	beliebiges Objekt duplizieren
roll	n Objekte verschieben
clear	alle Objekte vernichten
count	Objekte zählen
mark	Marke auf Stack legen
cleartomark	alle Objekte bis zur Marke vernichten
counttomark	Objekte bis zur Marke zählen

Betrachten Sie das folgende Beispiel:

```
1 1 60
{ ... Schleifenkoerper ...}
for
```

Der Operator *for* oben führt den Schleifenkörper 60 mal aus. Bei jedem Durchgang wird der Zähler (1 bis 60) auf den Operandenstack gelegt. Wie kann nun dieser Wert einer Variable zugewiesen werden?

```
{ /i exch def }
for
```

Während *exch* ein Stack-Operator ist, zählt *def* zu den Dictionary-Operatoren.

Was passiert dabei auf dem Stack? Stackzustand beim ersten Schleifendurchgang:

1
/i

Der Operator *exch* vertauscht nun die beiden Objekte 1 und /i auf dem Stack:

/i
1

Nun kann der Zähler der Variable i zugewiesen werden:

```
/i 1 def
```

Die Variable i erhält somit den Wert 1, beim zweiten Durchgang den Wert 2, usw. Noch eine Bemerkung zum Schrägstrich: die Variable i besitzt die Eigenschaft *literal* (gekennzeichnet durch den Schrägstrich), d.h. nicht-ausführbar, sie wird daher in den Stack gestellt.

Betrachten Sie im File effekt.eps die kleine Prozedur *crlf*, die eine Bewegung an den Anfang der nächsten Zeile ausführt:

```
/crlf   % carriage return, line feed
{ currentpoint 10 sub
  exch pop 0 exch moveto } def
```

Der Operator *currentpoint* hat keine Operanden, gibt jedoch die aktuellen Koordinaten X und Y auf den Operandenstack zurück. Der Operator *sub* rechnet nun die Subtraktion: Y - 10. Die Kombination *exch pop* sorgt dafür, dass der X-Wert zuoberst auf den Stack kommt, um dann vernichtet zu werden. Nehmen wir an, Y sei 100, dann würde schliesslich der folgende moveto-Befehl ausgeführt:

```
0 90 moveto
```

Nun kann die zusätzliche Prozedur prtstring eine einzelne Zeichenkette World-World-World anschreiben und die Prozedur crlf aufrufen.

Der grafische Status

Der grafische Status ist im Imaging-Modell von PostScript eine zentrale Idee. Das gleiche grafische Objekt kann -- je nach grafischem Status -- mit unterschiedlichen Attributen wiedergegeben werden, also in unterschiedlicher Farbe, Grösse, unterschiedlichem Font, mit unterschiedlicher Position, usw. Auch das Koordinatensystem, die Strichdicke und weitere Parameter zählen zum grafischen Status. Beispiele:

```
1 setlinewidth
/Helvetica 25 selectfont
```

Beide Befehle verändern den grafischen Status. Während das erste Beispiel die Strichstärke festlegt (1 Punkt), setzt das zweite Beispiel den aktuellen Font und die Schriftgrösse (Helvetica in 25 Punkt). Das folgende Beispiel beruht auf dem Font Courier:

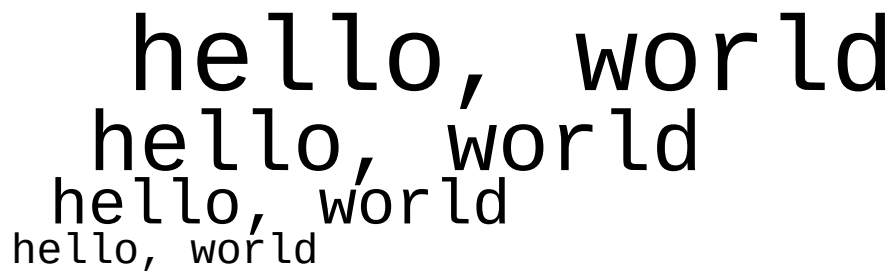

Abbildung 4: Courier in verschiedenen Schriftgrössen

Die Schriftgrösse ist ein Parameter des grafischen Status. Ein PostScript-Programm kann einen Font in jeder beliebigen Punktgrösse anwählen.

Das Beispiel

```
1 1 0 setrgbcolor % red-green-blue
```

bzw.

```
0 0 1 0 setcmykcolor % cyan-magenta-yellow-black
```

erzeugt die gelbe Farbe. Im ersten Fall befinden wir uns im RGB-Farbraum, im zweiten Fall im Zyan-Magenta-Gelb-Modell (mit zusätzlich Schwarz). Die Operanden geben die Farbanteile an; der Bereich geht jeweils von 0.0 bis 1.0. Die rote Farbe kann beispielsweise mit

```
1 0 0 setrgbcolor % rot
```

bzw.

```
0 1 1 0 setcmykcolor % rot = magenta + gelb
```

erzeugt werden.

Einige Operatoren zur Manipulation des grafischen Status:

selectfont	Font und Schriftgrösse festlegen (Level 2)
setlinewidth	Strichstärke festlegen
setdash	Strichmuster festlegen
setgray	Graustufe festlegen
setrgbcolor	Farbe festlegen, RGB-Modell
setcmykcolor	Farbe festlegen, CMYK-Modell
setscreen	Bildraster festlegen
settransfer	Transferfunktion für Graustufen festlegen (Bildkontrast)
sethalftone	Halbton-Dictionary einrichten (Level 2)

Um einen grafischen Status festzuhalten, verwenden PostScript-Programmierer häufig den Operator *gsave*. Mit *grestore* wird wieder derjenige grafische Zustand (aktuelle Position, Farbe, Schrift, etc.) hergestellt, der zum Zeitpunkt des gsave-Aufrufs galt. Beispiel:

```
0 200 moveto
(Hallo Welt) show % schwarz
```

```
gsave
  0 100 moveto
  1 0 0 setrgbcolor
  (Hallo Welt) show % rot
grestore

0 0 moveto
(Hallo Welt) show % schwarz
```

Der erste Schriftzug wird in Schwarz wiedergegeben, der zweite in Rot. Da der setrgbcolor-Operator von *gsave -- grestore* eingeklammert ist, druckt das Beispiel den dritten Schriftzug wieder in Schwarz.

Dictionary

Zusätzlich zu den herkömmlichen Konstrukten einer Programmiersprache unterstützt PostScript assoziative Tabellen (dictionaries), die ein kompaktes, übersichtliches und fehlerfreies Programmieren erlauben. Ein Dictionary ist eine Tabelle, deren Elemente Paare von PostScript-Objekten (Zahl, Zeichenkette, Name, Operator, etc.) sind. Das erste Element eines Paares nennt sich "Schlüssel”, das zweite Element "Wert”. Wenn nun der PostScript-Interpreter ein Namen-Objekt ausführen will, sucht er den Schlüssel zuerst im aktuellen Dictionary. Wird der Schlüssel nicht gefunden, sucht der Interpreter im nächsten Dictionary auf dem Dictionary Stack. Die Suche dauert so lange, bis der Schlüssel gefunden ist oder bis der Dictionary Stack kein weiteres Dictionary mehr enthält.

Wichtige Dictionary-Operatoren im Ueberblick:

def	einen Dictionary-Eintrag definieren
put	einen Schlüssel mit einem Wert verbinden
get	einen Wert aus einem bestimmten Dictionary entnehmen
load	nach einem Schlüssel suchen und Wert entnehmen
length	Anzahl Paare in einem bestimmten Dictionary
dict	neues Dictionary erzeugen
copy	ein Dictionary kopieren
undef	den Eintrag aus dem Dictionary entfernen (Level 2)

Wenn man kein bestimmtes Dictionary angibt, befindet man sich im sogenannten *userdict*. Beispiel:

```
/i 1 def
```

In diesem Beispiel haben wir der Variable i in userdict den Wert 1 zugewiesen.

Nun erhöhen wir den Wert von i um 1:

```
/i i 1 add def
```

Falls kein Dictionary angegeben wird, legt der PostScript-Server die Definitionen also ins userdict ab. Zentral ist das Dictionary-Konzept hinsichtlich Fonts, Muster, Bilder und Farbräume, weil dort wichtige Daten in Form von Dictionaries organisiert sind.

Das Zifferblatt der PostScript-Uhr (xclock)

Das abschliessende Beispiel zeigt ein vollständiges PostScript-Programm, welches ein Zifferblatt à la Salvador Dalì zeichnet:

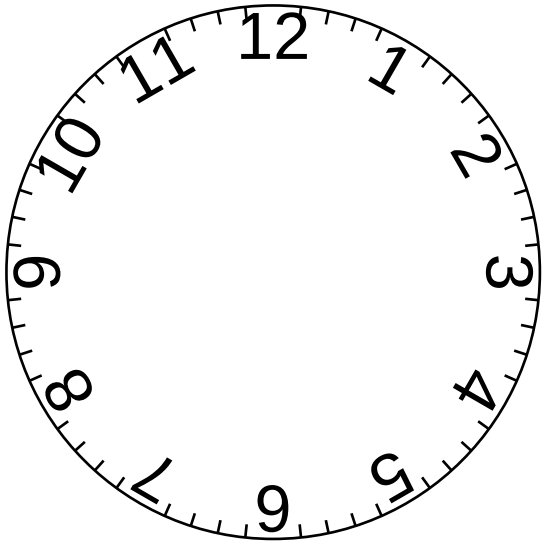

Abbildung 5: Zifferblatt der PostScript-Uhr à la Salvador Dalì (xclock)

Das Programm unten gibt zuerst die Bounding-Box an (im EPS-Kopf), zeichnet den Kreis und führt den Operator selectfont aus.

Dann folgt die for-Schleife, die von 1 bis 60 geht. Das Koordinatensystem wird bei jedem Schleifendurchgang um 6 Grad gedreht. Innerhalb der Schleife werden zwei Elemente gezeichnet:

1. Der Operator show schreibt die Stundenzahl an, wenn der Schleifenzähler durch 5 teilbar ist.
2. Der Operator stroke markiert jede Minute mit einem kleinen Strich.

Das vollständige Programmbeispiel -- Zifferblatt ohne die Uhrzeiger -- lautet wie folgt:

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: xclock.eps
%%BoundingBox: -101 -101 101 101
```

```
%%EndComments

1 setlinewidth
0 0 100 0 360 arc % Kreis
stroke

/Helvetica 25 selectfont

1 1 60 {       % Beginn der Schleife
/i exch def
-6 rotate
i 5 mod 0 eq % i durch 5 teilbar?
  { % Draw number
   i 5 idiv =string cvs dup stringwidth pop 2 div neg
      80 moveto
      show
  }
  { % Draw minute tick mark
      0 100 moveto
      0 5 neg rlineto
      stroke
  }
  ifelse
} for       % Ende der Schleife
% Ende des Programms
```

Bezüglich der Ablaufkontrolle enthält das obige Programmbeispiel den Operator *ifelse,* der hier zur Verdeutlichung wiederholt werden soll:

```
i 5 mod 0 eq
{ % Draw number
 ...  }
{ % Draw minute tick mark
 ...  }
ifelse
```

Wenn die Bedingung (i durch 5 teilbar) zutrifft, wird der Code "Draw Number” ausgeführt, andernfalls der Code "Draw minute tick mark”. Bei *mod* handelt es sich um einen sogenannten mathematischen Operator. Wichtige mathematische Operatoren sind:

add	Addition (zwei Werte zusammenzählen)
div	Division

idiv	Division, Integer
mul	Multiplikation
sub	Subtraktion
mod	Rest
abs	Absolutwert
neg	Negativwert
sqrt	Quadratwurzel
log	Logarithmus

Nun bleibt nur noch eine Zeile des Programms zu erklären:

```
i 5 idiv =string cvs dup stringwidth pop 2 div neg
```

Hier ist allerhand "verdrahtet”. Der mathematische Operator *idiv* teilt den Schleifenzähler (Minute 1 bis 60) durch fünf, berechnet also die Stunde (1 bis 12). Da der Stundenzähler angeschrieben werden soll, wandelt der Operator *cvs* den Stundenzähler, der sich jetzt auf dem Stack befindet, in eine Zeichenkette (=string) um, die anschliessend mit show ausgedruckt werden kann.

Und was soll stringwidth? Der Operator *stringwidth* nimmt sich die duplizierte Zeichenkette vom Operanden-Stack und stellt fest, wie gross diese ist. Der Operator stringwidth gibt zwei Werte auf den Stack, wovon uns nur die Breite interessiert (das andere Objekt wird mit *pop* vernichtet). Der Wert, der sich noch auf dem Stack befindet, wird durch zwei geteilt und negativ gemacht, damit der nachfolgende moveto-Operator eine Bewegung nach links ausführt.

Achtung: Falls Ihr PostScript-Interpreter den Operator selectfont nicht versteht, ersetzten Sie diesen durch die folgende Zeile:

```
/Helvetica findfont 25 scalefont setfont
```

Dies ist die alte Operator-Sequenz bei PostScript Level 1 (z.B. NeWS).

Wenn Sie das Zifferblatt auf Papier ausdrucken möchten, müssen Sie am Schluss des PostScript-Programms den Operator *showpage* anfügen, der dafür sorgt, dass das Papierblatt ausgegeben wird.

Literatur

Adobe Systems Inc. (1990),
PostScript Language, Reference Manual, 2nd Edition,
Addison-Wesley.

Söker, Wilfried (1992),
PostScript, Eine umfassende Einführung,
Vieweg.

Vollenweider, Peter (1991),
PostScript -- Konzeption und Anwendungen, 2. Auflage,
Carl Hanser Verlag.