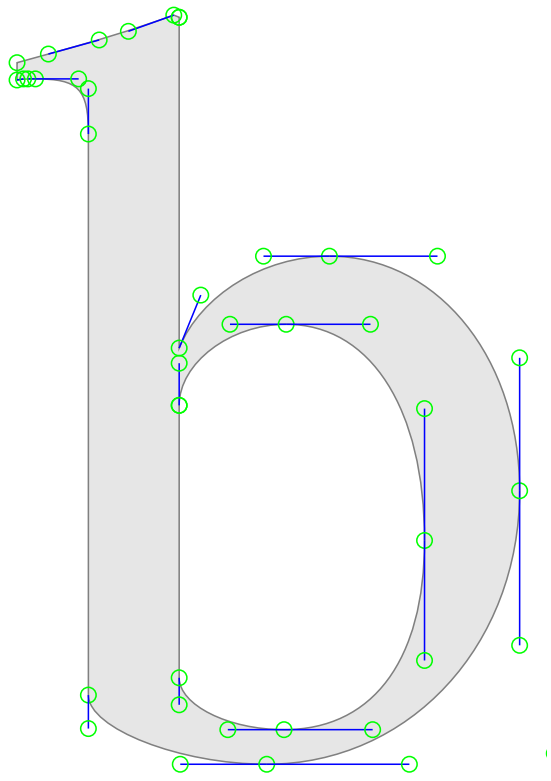


- Grundsätzlich werden Schnitte einzelner Schriftzeichen auf Basis von Kurven konstruiert – genauso wie die bisherigen geometrischen Figuren auch.
- Eine Reihe von Figuren für Schriftzeichen bilden einen Schriftschnitt, der in speziellen assoziativen Arrays (*dictionaries*) zusammengefasst wird, der einigen speziellen Konventionen genügt.
- PostScript bietet eine Vielzahl von speziellen Operatoren an, die mit Schriftschnitten und Figuren für Schriftzeichen umgeht.
- Dies erlaubt es, Schriftschnitte besonders effizient zu implementieren (z.B. durch die Verwendung von Caches).

hello.ps

```
/Times-Roman findfont 12 scalefont setfont
50 700 moveto
(Hello World!) show
showpage
```

- `findfont` sucht nach dem genannten Schriftschnitt und lädt das zugehörige assoziative Array auf den Stack.
- Mit `12 scalefont` wird der Schriftschnitt entsprechend skaliert. Das wäre prinzipiell auch mit dem `scale`-Operator machbar. `scalefont` bezieht sich aber nur auf den einen Schriftschnitt und nicht auf die übrigen Pfade.
- Mit `setfont` wird der oben auf dem Stack liegende Schriftschnitt zum aktuellen Schriftschnitt. Es gibt hierfür keine Voreinstellung!
- `show` erwartet eine Zeichenkette und stellt diese mit dem aktuellen Schriftschnitt an der aktuellen Position (die wohldefiniert sein muss) dar.

Standardmäßig gehören die drei Schriftfamilien Times-Roman, Helvetica und Courier zu den unter PostScript verfügbaren Schriftschnitten:

Times-Roman	Helvetica	Courier
Times-Italic	*Helvetica-Oblique*	*Courier-Oblique*
Times-Bold	**Helvetica-Bold**	**Courier-Bold**
Times-BoldItalic	***Helvetica-BoldOblique***	***Courier-BoldOblique***

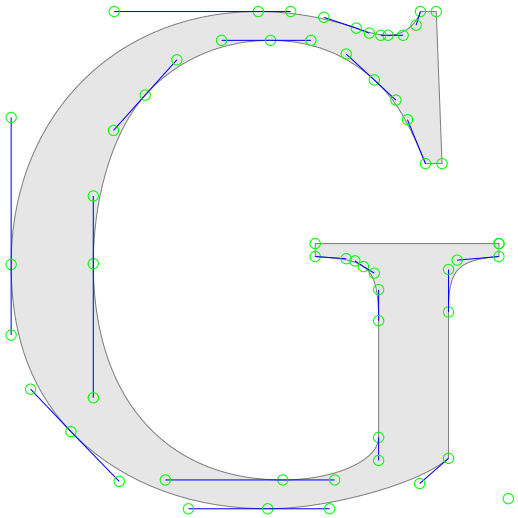
Hinzu kommt noch ein Schriftschnitt für diverse (insbesondere mathematische) Symbole. Optional stehen typischerweise zahlreiche weitere Schriftschnitte zur Verfügung.

Ivan Tschichold, 1928:

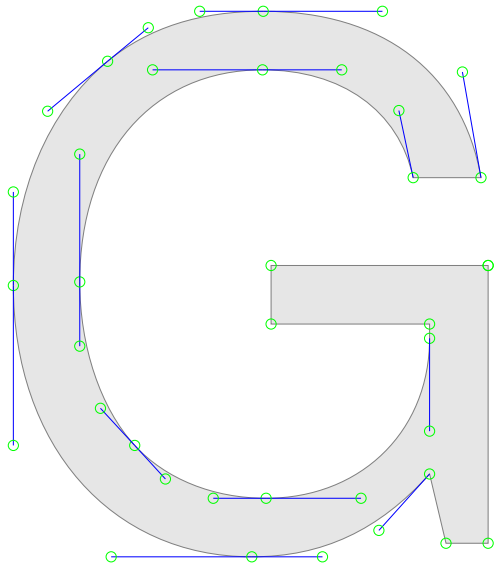
> *Der Mensch des 15. Jahrhunderts stand aufrecht vor dem Lesepult und las sich oder anderen den Inhalt laut vor. Daher die großen Lettern, die für die Mehrzahl der gotischen Bücher charakteristisch sind. Erst die zunehmende Beschleunigung des Lesetempos machte in der Folgezeit die Verwendung kleinerer Typen möglich und notwendig. Das laute und langsame Lesen, das „Abtasten“ des Einzelbuchstabens, des Einzelworts, ist in unserer Zeit dem Überfliegen des Textes gewichen. Die Lesetechnik des heutigen Menschen erzeugte die spezifische Form des Zeitungssatzes [...] Die optische Erscheinung der Zeitung gibt ein Sinnbild des heutigen Lebenstempos.*

(zitiert nach *Schrift und Typografie* von Stefan Waidmann)

Times Roman

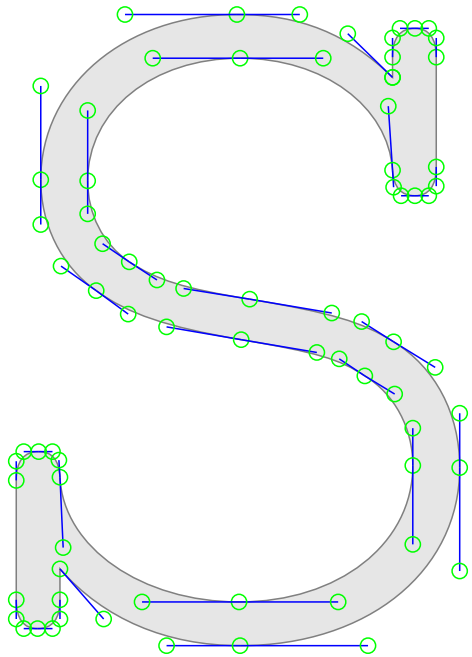


- In einer Beilage der *Times* über den Buchdruck erschien am 29. Oktober 1929 ein Artikel von Stanley Morrison unter dem Titel *Newspaper Types: A Study of The Times*, in dem die Zeitungstypografie heftig kritisiert wurde.
- Das nahm die Times zum Anlass, Morrison mit dem Entwurf eines neuen Schriftschnitts zu beauftragen. 1931 war Morrison mit seiner Arbeit fertig. Der Schriftschnitt wurde von Monotype (zuerst in 9 Punkt) implementiert.
- Seit dem 3. Oktober 1932 bis heute (abgesehen von einer kurzen Unterbrechung) verwendet die Times diese Schriftfamilie.
- Sie wurde aber zwischenzeitlich mehrfach überarbeitet.



- Bereits zu Beginn des 19. Jahrhunderts gab es serifenlose Schriften. Prominentes Beispiel ist die Akzidenz Grotesk, die 1898 bei der Berthold AG in Berlin erschien.
- Im 20. Jahrhundert stießen bei der Suche nach elementaren Formen und einer neuen Sachlichkeit diese Schriftschnitte auf Interesse. Das galt insbesondere für das Bauhaus (hier entwarf Paul Renner 1925 die Futura) und später in den 40-er und 50-er Jahren für Schweizer Typografen.
- Die Helvetica entstand als Überarbeitung der Akzidenz Grotesk in der Schriftgießerei Haas in Basel. Die Schrift wurde von Max Miedinger nach Vorgaben von Eduard Hoffmann entwickelt und erschien 1957.

Courier



- Courier wurde für Schreibmaschinen 1952 von Howard Kettler im Auftrag von IBM entworfen.
- Courier gewann recht rasch auch Popularität bei anderen Schreibmaschinen-Herstellern.
- Analog zu Schreibmaschinen sind alle Zeichen gleich weit.

PathLoad	/usr/local/share/ghostscript/fonts/n019003l.pfb
FID	--nostringval--
FontBBox	[-174 -285 1028 953]
Private	dictionary with 15 elements
CharStrings	dictionary with 560 elements
FontType	1
FontInfo	dictionary with 10 elements
Encoding	array with 256 elements
FontName	Helvetica
FontMatrix	[0.001 0 0 0.001 0 0]
.OrigFont	dictionary with 14 elements
.Alias	Helvetica
PaintType	0
UniqueID	5020902

- Ein Schriftschnitt ist in PostScript ein assoziatives Array mit Einträgen, die vorgegebenen Konventionen entsprechen.

showfonts.eps

```
% dict DisplayDict
/DisplayDict {
   6 dict begin
      /d exch def
      /Helvetica findfont 12 scalefont setfont
      /x 10 def
      /htab 100 def
      /y 225 def
      d {
         /value exch def
         /key exch def
         x y moveto
         key ToString show
         x htab add y moveto value ToString show
         /y y 16 sub def
      } forall
   end
} def

/Helvetica findfont DisplayDict
```

showfonts.eps

```
/d exch def
/Helvetica findfont 12 scalefont setfont
/x 10 def
/htab 100 def
/y 225 def
d {
   /value exch def
   /key exch def
   x y moveto
   key ToString show
   x htab add y moveto value ToString show
   /y y 16 sub def
} forall
```

- `DisplayDict` durchläuft mit `forall` alle Schlüssel/Werte-Paare des übergebenen assoziativen Arrays und gibt sowohl den Schlüssel als auch den Wert aus, nachdem sie zuvor mit `ToString` typabhängig in eine Zeichenkette konvertiert worden sind.

Schlüssel	Typ	Bedeutung
FontType	integer	Art der Spezifikation des Schriftschnitts
FontMatrix	array	Transformations-Matrix für die Kurven innerhalb der Definitionen für die Zeichen; typisch ist ein 1000x1000-Koordinatensystem
FontInfo	dict	Array mit weiteren Feldern, die den Schriftschnitt beschreiben
Encoding	dict	bildet Werte aus dem Bereich $[0, 255]$ in Namen für die einzelnen Zeichen ab
FontBBox	array	Bounding-Box aller übereinander gezeichneter Zeichen
CharStrings	dict	spezielle Repräsentierungen der einzelnen Zeichen bei Type-1 Schriftschnitten

Der `FontType` wählt eine der drei folgenden Spezifikationsarten aus:

Type 0	Zusammengesetzter Schriftschnitt, der auf anderen Schriftschnitten basiert
Type 2	Besteht aus speziell kodierten Prozeduren für die einzelnen Zeichen, die dem *Adobe Type 1 Font Format* entsprechen
Type 3	Alle Prozeduren für die Zeichen sind reguläre PostScript-Prozeduren

Type 1 Schriftschnitte sind im Vergleich zu Type 3 kompakter, effizienter und haben optional zusätzliche Hinweise zur optimalen Darstellung in Abhängigkeit der Rasterung und der gewählten Schriftgröße
Weitere Typen sind bei Level 3 hinzugekommen wie etwa Type 42 für TrueType-Schriftschnitte.

Definition eines Type-3-Schriftschnitts

torniello.eps

```
% Generierung eines Type-3-Fonts mit den Buchstaben
% E, T und X von Francesco Torniello:
% name Torniello font
/Torniello {
   2 dict begin
      /name exch def % unter diesem Namen wird der Font registriert
      /newfont 8 dict def
      newfont begin
         % Definition der einzelnen Eintraege ...
      end
      name newfont definefont
   end
} def

% Demo
/TorniellosFont Torniello 100 scalefont 0 0 moveto setfont (TEX) show
```

- Type-3-Schriftschnitte lassen sich mit regulären PostScript-Anweisungen erstellen.
- Im wesentlichen ist ein assoziatives Array entsprechend den Konventionen richtig zu füllen und mit dem Operator `definefont` in einen Schriftschnitt zu konvertieren.

Definition eines Type-3-Schriftschnitts

torniello.eps

```
newfont begin
   /FontType 3 def  % Font, der in PostScript definiert ist
   /FontMatrix [.001 0 0 .001 0 0] def
   /FontBBox [-30 0 950 950] def
   /Encoding 256 array def
   0 1 255 {
      Encoding exch /.notdef put
   } for
   Encoding 69 /E put
   Encoding 84 /T put
   Encoding 88 /X put
   % ... weitere Eintraege ...
end
```

- Assoziative Arrays lassen sich relativ elegant füllen in einer entsprechenden Klammerung mit `begin` und `end`.
- Die `FontMatrix` entspricht derjenigen, die mit
  `1 1000 div dup matrix scale`
  erzeugt werden würde.
- Die Operatoren `get` und `put` erlauben einen indizierten Zugriff auf Arrays, assoziative Arrays und Strings.

torniello.eps

```
% Tabelle der Weiten der einzelnen Buchstaben
/CharWidth 3 dict def
CharWidth begin
   /E 700 def
   /T 850 def
   /X 950 def
end
```

- Für jedes einzelne Zeichen ist die Weite in einer Tabelle festzulegen.
- Diese Weite gibt an, wieviel Platz für ein Zeichen reserviert wird. Das Zeichen kann weniger Platz benötigen oder auch darüber hinausragen – mit der Gefahr, dass sich dann die Darstellung mit derer anderer Zeichen überschneidet.

torniello.eps

```
% Bounding-Boxes der einzelnen Buchstaben
/CharBB 3 dict def
CharBB begin
   /E [ 100 0 700 900 ] def
   /T [ 0 0 900 950 ] def
   /X [ -30 0 950 900 ] def
end
```

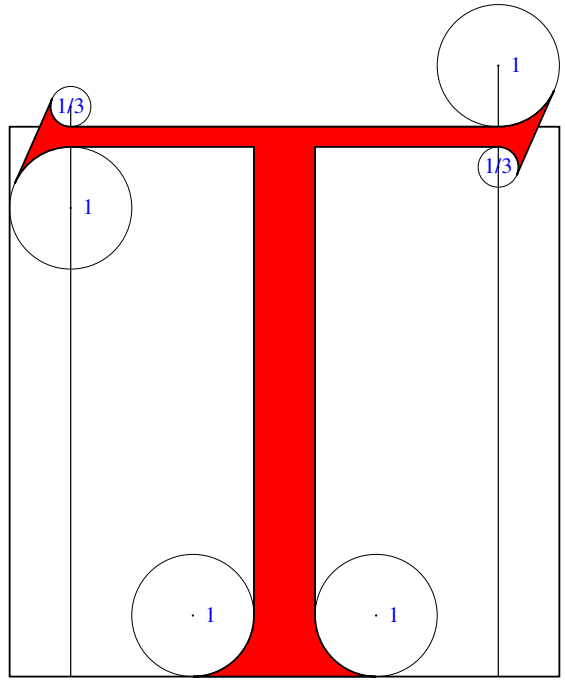
- Ferner ist es ratsam, für jedes einzelne Zeichen die Bounding-Box einzugrenzen.
- Diese Bounding-Box wird für das Caching verwendet, d.h. Teile des Zeichens, die auf diese Weise abgeschnitten werden, können mal gezeichnet werden oder auch mal entfallen.
- Je enger die Bounding-Box ist, umso effizienter kann PostScript damit arbeiten und benötigt dann auch dafür weniger Speicherplatz im Cache.

- Die Weite wird verwendet, wenn es um das Aneinanderreihen von Schriftzeichen geht.
- Die Bounding-Box legt nur eine obere Schranke fest, in dem das Schriftzeichen liegt. Dies ist nur für das Caching relevant.
- Die roten Schachteln machen hier die Weite deutlich. Zu erkennen ist, dass das »T« und das »X« etwas über die angegebenen Weiten herausragen.

torniello.eps

```
% Zeichenprozeduren fuer die einzelnen Buchstaben
/CharProcs 4 dict def
CharProcs begin
   /.notdef {} def
   % ... die einzelnen Zeichen ...
end
```

- Das assoziative Array `CharProcs` enthält für jedes Zeichen eine Prozedur, die dieses zeichnet.



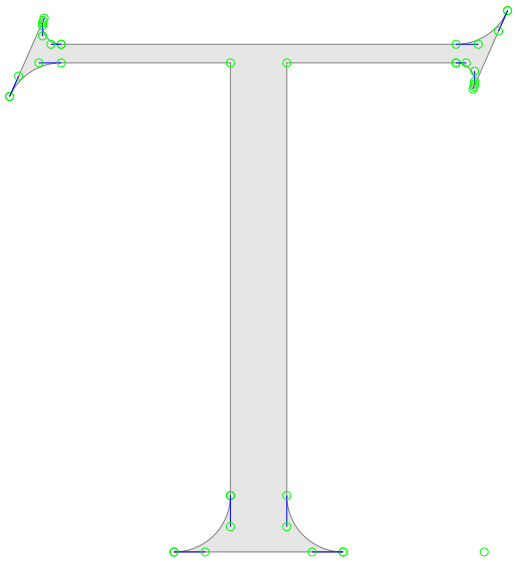
Spezifikation nach Torniello:

Der Buchstabe T wird aus dem Quadrat geformt. Zunächst wird der Schaft, der ein Punkt breit ist, in die Mitte des Quadrats platziert mit den Kreisen an der Grundlinie, so wie Du sie siehst. Danach beginne einen Punkt innerhalb der Ecke oben links und zeichne entlang der oberen Horizontalen, einen Punkt vor der rechten Ecke endend, und füge die Kreise wie gezeigt hinzu. Dann zeichne eine weitere horizontale Linie innerhalb des Quadrats ein Drittel eines Punktes von der zuvor gezeichneten Linie entfernt und von der gleichen Länge, und binde sie in die Kreise ein, die Du eingezeichnet siehst.

torniello.eps

```
/T {
   newpath
   300 0 moveto
   600 0 lineto
   600 100 100 270 180 arcn
   500 867 lineto
   800 867 lineto
   800 834 33 90 336.1956690 arcn
   800 150 2563 sqrt mul 83 div add 79650 83 div lineto
   800 1000 100 336.1956690 270 arcn
   100 900 lineto
   100 933 33 270 156.1956690 arcn
   100 150 2563 sqrt mul 83 div sub 67011 83 div lineto
   100 767 100 156.1956690 90 arcn
   400 867 lineto
   400 100 lineto
   300 100 100 0 270 arcn
   closepath
   fill
} def
```

- Dies ist die Zeichenprozedur für das »T«.



- An dieser aus der internen Repräsentierung der Figur abgeleiteten Darstellung des »T« lässt sich erkennen, wie die Bögen in Bézier-Kurven konvertiert worden sind.
- Der isolierte Punkt ganz rechts unten ergibt sich aus einem impliziten `moveto` ganz am Ende.

Definition eines Type-3-Schriftschnitts

torniello.eps

```
% font charname BuildGlyph
/BuildGlyph {
   3 dict begin
      /charname exch def
      /font exch def
      /cw font /CharWidth get def
      /cbb font /CharBB get def
      cw charname get 0 % Weite
      cbb charname get aload pop % Bounding-Box
      setcachedevice
      /cp font /CharProcs get def
      cp charname known not {
         /charname /.notdef def
      } if
      cp charname get exec
   end
} bind def
```

- `BuildGlyph` wird für jedes zu zeichnende Zeichen, das noch nicht im Cache zur Verfügung steht, aufgerufen.
- `setcachedevice` sorgt dafür, dass das Zeichen simultan im Cache und in der tatsächlichen Ausgabe landet.

torniello.eps

```
/BuildChar {
   1 index /Encoding get exch get
   1 index /BuildGlyph get exec
} bind def
```

- `BuildChar` wird nur von älteren Level-1-Interpretern verwendet und kann auf Basis von `BuildGlyph` formuliert werden.

TEX

torniello.eps

```
/TorniellosFont Torniello 100 scalefont 0 0 moveto setfont (TEX) show
```

- Nach der Definition kann der Schriftschnitt sofort verwendet werden.
- Schriftschnitte des Typs 3 haben jedoch einige Schwächen (z.B. in Bezug auf die Rasterung) und erfreuen sich keiner großen Unterstützung, so dass sie immer seltener eingesetzt werden.

uphill.eps

```
/Helvetica-ISO findfont 20 scalefont setfont

20 20 moveto
60 rotate
(Es geht aufwärts!) show
```

- Die Transformations-Operatoren `rotate`, `scale` und `translate` lassen sich auch auf Zeichen aus Schriftschnitten anwenden.

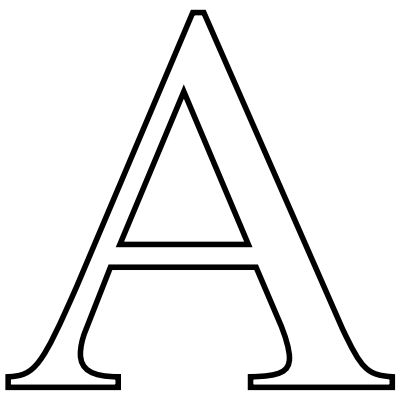
uphill.eps

```
/Helvetica-ISO /Helvetica findfont MakeISOFont pop
```

- Per Voreinstellung enthält der Encoding-Vektor der vorgegebenen Type 1 Schriftschnitte nur ASCII-Zeichen, selbst wenn der Schriftschnitt auch Zeichen aus ISO-8859-1 unterstützt.
- Der Standard-Vektor `Encoding` lässt sich aber durch den Vektor `ISOLatin1Encoding` austauschen.
- `MakeISOFont` macht genau dieses auf einer Kopie. (Letzteres ist zwingend notwendig.)
- Zur Unterstützung asiatischer Schriftschnitte gibt auch fortgeschrittene Kodierungstechniken, die auch mit Byte-Folgen operieren können.

uphill.eps

```
% name font MakeISOFont font
/MakeISOFont {
   3 dict begin
      /font exch def
      /name exch def
      /newfont font length dict def
      newfont begin
         font {
            % copy all entries with the exception of /FID
            1 index /FID ne {
               def % copy
            } {
               pop pop % discard
            } ifelse
         } forall
         /Encoding ISOLatin1Encoding def
      end
      name newfont definefont
   end
} def
```

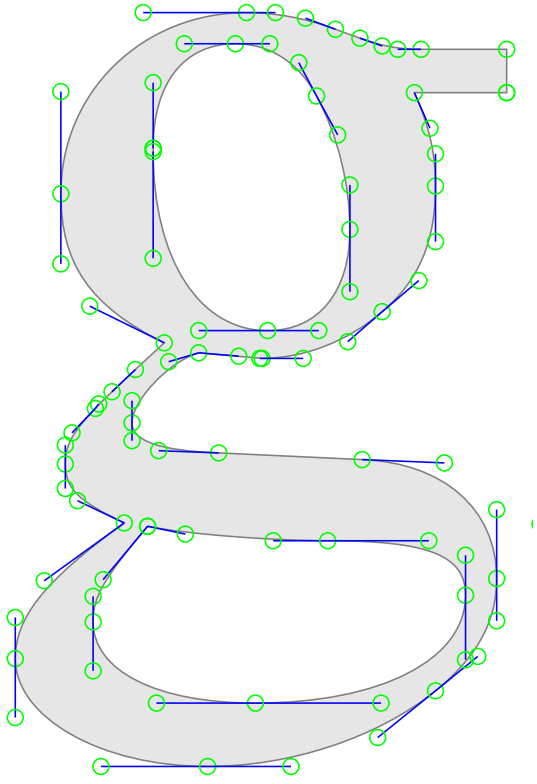


strokedA.eps

```
/Times-Roman findfont 100 scalefont setfont
10 10 moveto
(A) false charpath stroke
```

- Der Operator `charpath` zeichnet den gegebenen String nicht, sondern ergänzt den aktuellen Pfad um die Pfade der einzelnen Zeichen in dem aktuellen Schriftschnitt.
- Der Boolean-Operator sollte gesetzt werden in Abhängigkeit von der Folge-Operation für den Pfad:
  - `false` für `stroke`
  - `true` für `fill` und `clip`

Iteration durch einen Pfad



glyph-outline.eps

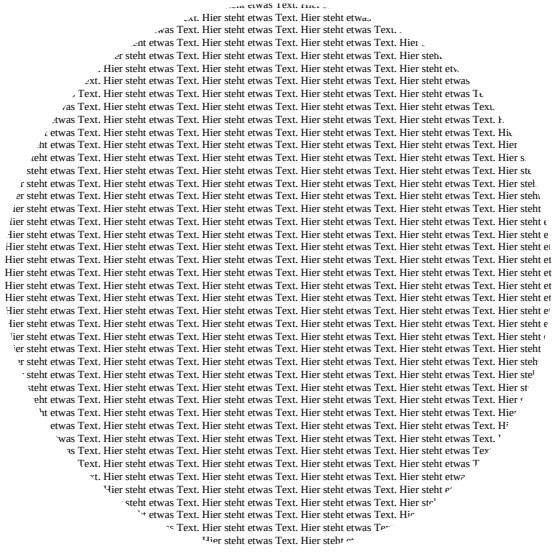
```
newpath 0 0 moveto (g) true charpath
% x y moveto -
{
    % ...
}
% x y lineto -
{
    % ...
}
% x1 y1 x2 y2 x3 y3 curveto -
{
    % ...
}
% - closepath -
{
    % ...
}
pathforall
```

- Mit `pathforall` ist es möglich, durch alle Komponenten eines Pfads durchzuiterieren.

Iteration durch einen Pfad

glyph-outline.eps

```
newpath 0 0 moveto ch true charpath
% x y moveto -
{
   /y exch def /x exch def
   x y MarkPoint x y moveto
}
% x y lineto -
{
   /y exch def /x exch def
   x y MarkPoint x y lineto
}
% x1 y1 x2 y2 x3 y3 curveto -
{
   /y3 exch def /x3 exch def
   /y2 exch def /x2 exch def
   /y1 exch def /x1 exch def
   currentpoint /y exch def /x exch def
   gsave
      newpath
      x y moveto x1 y1 lineto
      x2 y2 moveto x3 y3 lineto
      0 0 1 setrgbcolor 1 setlinewidth stroke
   grestore
   x1 y1 MarkPoint x2 y2 MarkPoint x3 y3 MarkPoint
   x1 y1 x2 y2 x3 y3 curveto
}
% - closepath -
{ closepath }
pathforall
```

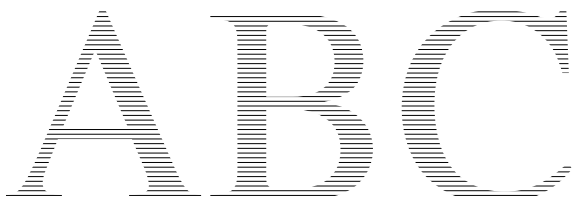


circletext.eps

```
300 300 300 0 360 arc clip

/Times-Roman findfont 12 scalefont setfont
595 -14 0 {
   /y exch def
   0 y moveto
   6 { (Hier steht etwas Text. ) show } repeat
} for
```

- Mit `clip` wird der sogenannte Clipping-Pfad definiert.
- Außerhalb der Füllfläche des Clipping-Pfades wird nicht gezeichnet.
- Auf diese Weise können unerwünschte Teile einer Zeichnung weggeblendet werden.



linedabc.eps

```
/Times-Roman findfont 60 scalefont setfont
0 0 moveto
(ABC) true charpath clip
clippath pathbbox
/ury exch cvi def
/urx exch cvi def
/lly exch cvi def
/llx exch cvi def

newpath
lly 1 ury {
   dup llx exch moveto
   urx exch lineto
} for
0.2 setlinewidth stroke
```

- Auch die Umrisse von Zeichen können als Clipping-Pfad verwendet werden.

linedabc.eps

```
clippath pathbbox
/ury exch cvi def
/urx exch cvi def
/lly exch cvi def
/llx exch cvi def
```

- `clippath` macht den Clipping-Pfad zum aktuellen Pfad.
- `pathbbox` liefert die Bounding-Box des aktuellen Pfades in Form der vier Zahlen, die hier allerdings Gleitkommazahlen sein können.
- Durch das Ausmessen des Schriftzuges wissen wir, in welchem Bereich es sich überhaupt lohnt, Linien zu zeichnen.