Computergrafik-Übungen
W. Kurth, E. Roth
WS 2001/02

# Einführungskurs PostScript, Teil 3

## Umrisslinien von Buchstaben

Der Befehl **`charpath`** fügt die Umrisslinien der Zeichen des als Argument übergebenen Strings an den aktuellen Pfad an.
2 Argumente
1. Argument: String
2. Argument: Boolean
wenn 2. Arg. **`false`**: Aufbereitung für anschließenden **`stroke`**-Befehl
wenn 2. Arg. **`true`**: Aufbereitung für anschließenden **`fill`**-Befehl
(2. Argument nur relevant für bestimmten Typ von Fonts, sog. stroked-Fonts)
Vor Aufruf von **`charpath`** muss ein Font aktiviert sein!

Beispiel:

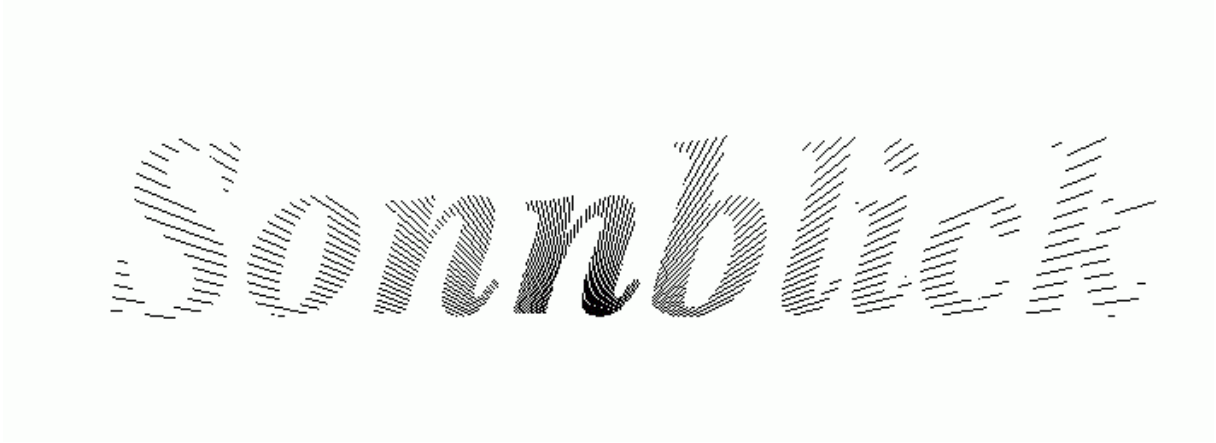
```
/Times-Bold findfont 40 scalefont setfont
100 600 moveto
(Grauer Text) true charpath
gsave
0.6 setgray fill
grestore
stroke
showpage
```

Ergebnis:

Die Verwendung der Umrisslinie ist auch zum Clipping möglich:

```
/Strahlen
   {
   0 1.5 360
      {
      gsave
      rotate
      0 0 moveto
      200 0 rlineto
      stroke
      grestore
      } for
   } def
/Times-BoldItalic findfont 64 scalefont setfont
200 600 translate   % Ursprung jetzt oben auf der Seite
0.24 setlinewidth
0 0 moveto
(Sonnblick) true charpath clip
newpath
120 -15 translate   % verschiebe Ursprung etwa in die
                    % Mitte des Clipping-Umrisses
Strahlen
showpage
```

Ergebnis:



Transformationsmatrizen in PostScript

PostScript ermöglicht die explizite Ein- und Ausgabe von Koordinaten-Transformationsmatrizen in homogenen Koordinaten, d.h. als 3×3-Matrizen. Darin sind 3 Einträge (die unterste Zeile) fest vorbelegt mit ( 0 0 1 ).
Matrix-Notation in PostScript: Array mit 6 Einträgen.

Matrix $\begin{pmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{pmatrix}$
in PostScript als **`[ a b c d e f ]`**.

Erste vier Einträge: kartesische Transformationsmatrix (linearer Anteil der affinen Abb.)
letzte zwei Einträge: Translationsanteil.

Der Befehl **`concat`** bewirkt die Multiplikation (Konkatenation) der als Argument übergebenen Transformationsmatrix mit der aktuellen Transformationsmatrix (CTM, *current transformation matrix*). Die als Argument übergebene Matrix wird dabei vom Stack genommen.

Beispiele:

Wir wenden probeweise die folgenden Transformationsmatrizen an:

$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ Streckung um Faktor 2 in *y*-Richtung

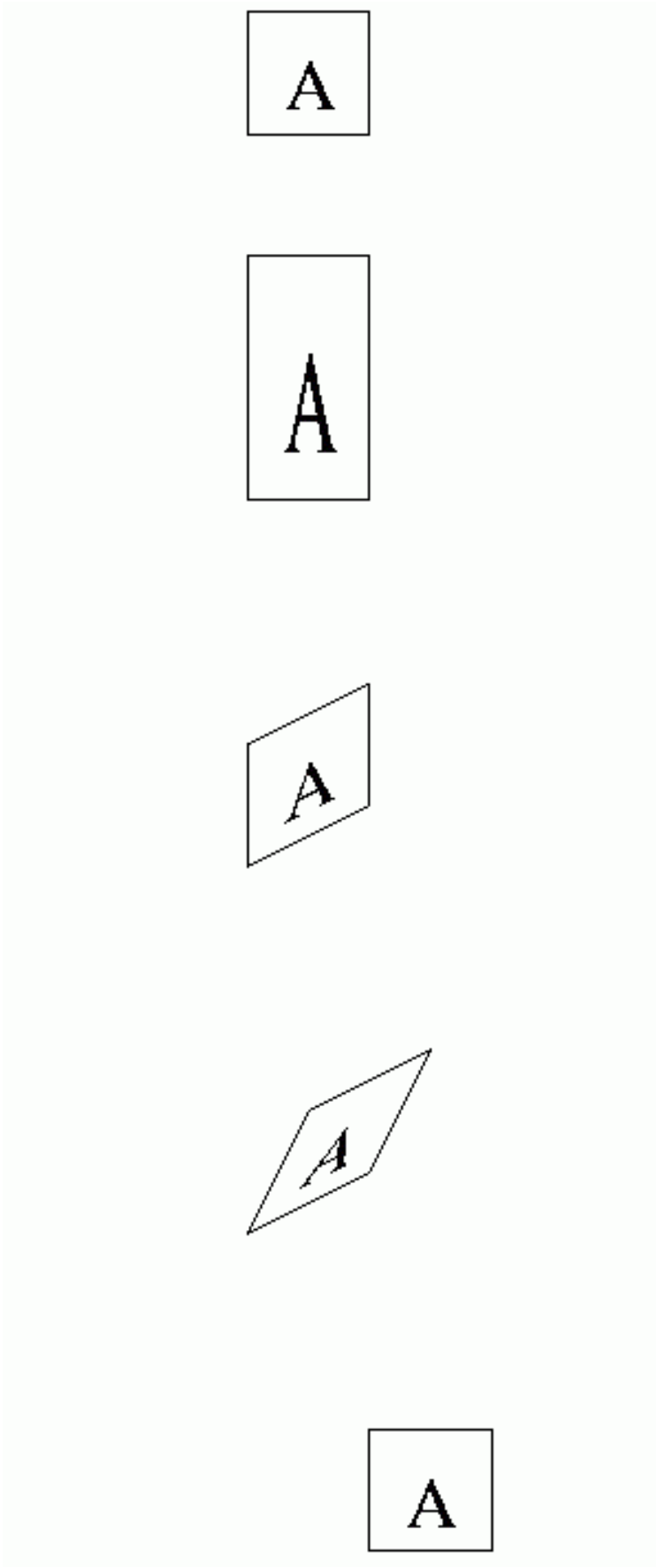
$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ Scherung entl. der *y*-Achse um Scherungsbetrag 0,5

$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ Abb., die eine Verzerrung entl. beider Achsen bewirkt

$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ Translation um (50; 20).

```
/ARahmen
   {
   /Times findfont 30 scalefont setfont
   newpath
   0 0 moveto
   0.2 setlinewidth
   50 0 rlineto
   0 50 rlineto
   -50 0 rlineto
   closepath
   stroke
   15 10 moveto
   (A) show
   } def
200 700 translate
ARahmen
0 -150 translate
gsave
[ 1 0 0 2 0 0 ] concat % Streckung in y-Richtung
ARahmen
grestore
0 -150 translate
gsave
[ 1 0.5 0 1 0 0 ] concat % Scherung entl. y-Achse
ARahmen
grestore
0 -150 translate
gsave
[ 1 0.5 0.5 1 0 0 ] concat % Verzerrung, beide Achsen
ARahmen
grestore
0 -150 translate
[ 1 0 0 1 50 20 ] concat % Translation
ARahmen
showpage
```

Ergebnis:



Weitere Befehle für Transformationsmatrizen:
(*kursiv*: Stackeinträge vor / nach der Anwendung des Operators)

**`matrix`** → *Matrix*
Erzeugen einer leeren Matrix

*Matrix* **`identmatrix`** → *Matrix*
die Matrix wird mit der Einheitsmatrix überschrieben

*Matrix* **`defaultmatrix`** → *Matrix*
die Matrix wird mit der Standard-Transformationsmatrix (für die Umrechnung auf Gerätekoordinaten) überschrieben

*Matrix* **`currentmatrix`** → *Matrix*
die Matrix wird mit der aktuellen Transformationsmatrix (CTM) überschrieben

*Matrix1 Matrix2 Matrix3* **`concatmatrix`** → *Matrix3*
Matrix3 wird mit dem Produkt der Matrizen 1 und 2 überschrieben

*Matrix1 Matrix2* **`invertmatrix`** → *Matrix2*
Matrix2 wird mit der inversen Matrix zu Matrix1 überschrieben

*Matrix* **`setmatrix`** →
die Matrix wird direkt als aktuelle CTM übernommen

Varianten der Befehle **`translate`**, **`scale`**, **`rotate`**: Die dem Befehl entsprechende Transformationsmatrix wird in die mit übergebene Matrix geschrieben.
*tx ty Matrix* **`translate`** → *Matrix*
*sx sy Matrix* **`scale`** → *Matrix*
*Winkel Matrix* **`rotate`** → *Matrix*

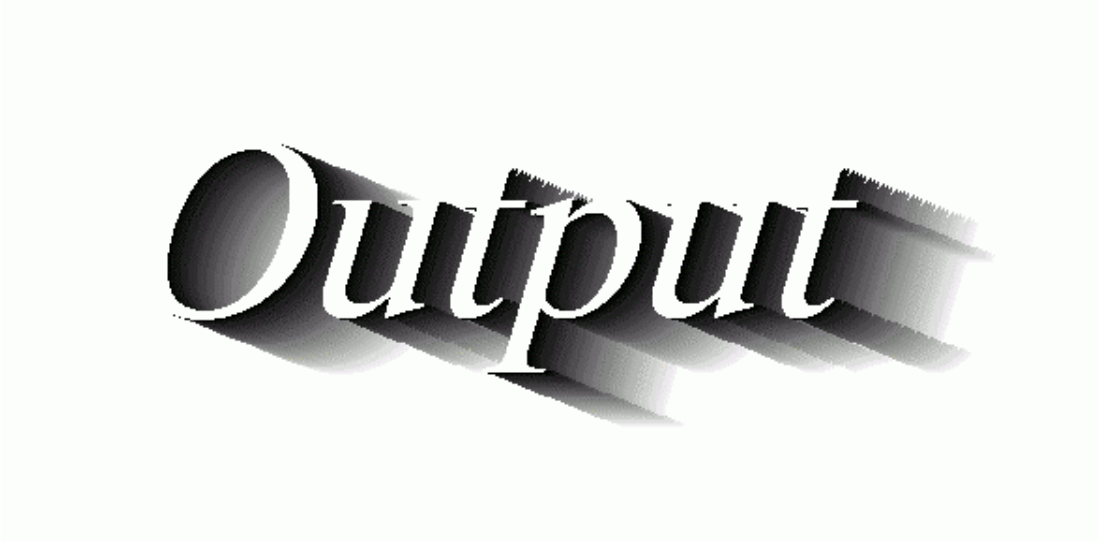
Die Transformationsmatrix wird auch auf Schrift angewandt. Beispielsweise können wir die folgende Scherungsmatrix verwenden, um kursive Schrift ohne Verwendung eines Kursiv-Fonts zu erzeugen:

$$\begin{pmatrix} 1 & 0.3 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(entspr. der PostScript-Notation **[ 1 0 0.3 1 0 0 ]** ).

```
/Times findfont 60 scalefont setfont
/PrintOutput
   {
   0 0 moveto
   gsave
   [ 1 0 0.3 1 0 0 ] concat
   (Output) show
   grestore
   } def
100 700 translate
0.95 -0.03 0
   {
   setgray
   PrintOutput
   -1 0.4 translate
   } for
1 setgray PrintOutput
showpage
```

Ergebnis:




Die mit **`concat`** veränderte Transformationsmatrix wirkt auf alle Objekte (Schrift und Zeichnungen). Wenn eine Transformation nur für Schrift gewünscht wird, bietet sich der Befehl **`makefont`** an, der die direkte Angabe einer Transformationsmatrix für einen Font ermöglicht.
Im obigen Beispiel wäre die **`setfont`**-Zeile (1. Zeile) zu ersetzen durch:

```
/Times findfont [ 1 0 0.3 1 0 0 ] makefont
60 scalefont setfont
```

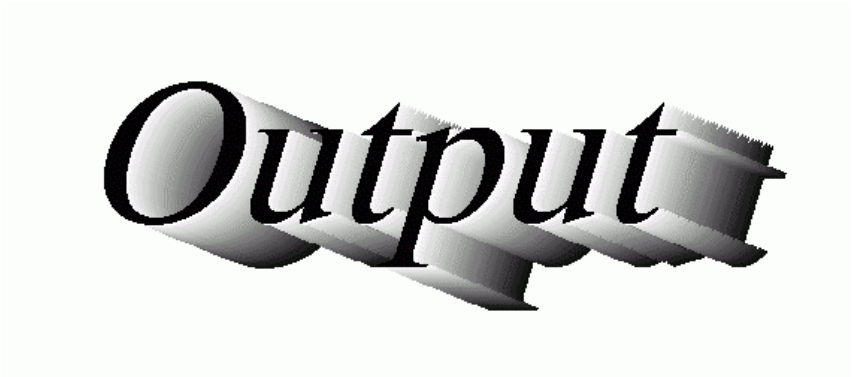
Auf das **`concat`** ist dann zu verzichten.
Das Ergebnis ist dasselbe wie oben.

Transferfunktion für Grauwerte

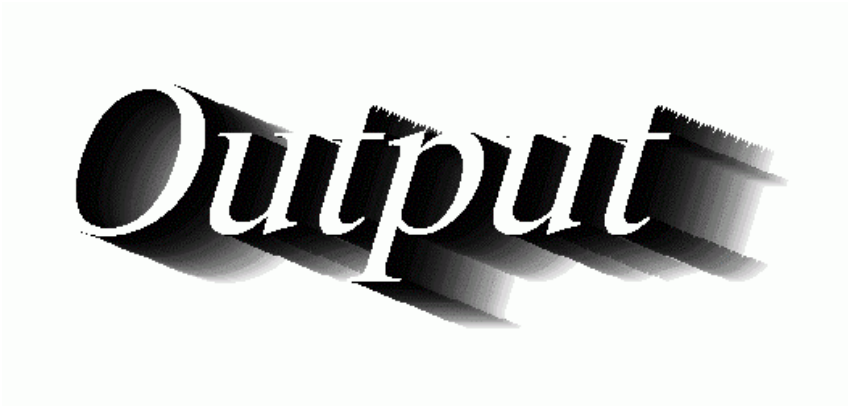
Der Standard-Graukeil (siehe Übungsblatt 1, Aufg. 4) liefert, je nach Ausgabegerät, oft keine befriedigenden Resultate. Durch Angabe einer Transferfunktion mit Argumenten und Werten zwischen 0 und 1 kann der Verlauf der Grauwerte einer optisch befriedigenden, gleichmäßigen Abfolge angenähert werden:

Der Befehl settransfer erwartet als Argument eine Prozedur, die die Berechnung der Transferfunktion durchführt.

Beispielsweise können wir im obigen Beispiel die Funktion (1–*x*) als Transferfunktion definieren und damit die Grauwerte invertieren:
Einfügen der Zeile
**`{ 1 exch sub } settransfer`**
nach der Prozedurdefinition für PrintOutput liefert:




Anderes Beispiel: die Funktion $x^2$
**`{ dup mul } settransfer`**



Explizite Gerätekoordinaten-Berechnung

Befehl für die Umrechnung von Welt- nach Geräte-Koordinaten:

*x y* **transform** → *x' y'*

Umkehrung:

*x' y'* **itransform** → *x y*

Anwendung: Vermeidung von Aliasing-Effekten bei Linienstärken und Linienabständen durch Justieren der Weltkoordinaten auf ganzzahlige Gerätekoordinaten.

Beispiel:

```
100 200 moveto
300 0 rlineto stroke
```

erzeugt eine Linie, deren Anfangspunkt in Gerätekoordinaten exakt oder gerundet sein kann ⇒ Strichdicke kann je nach Lage des Anfangspunktes leicht variieren.

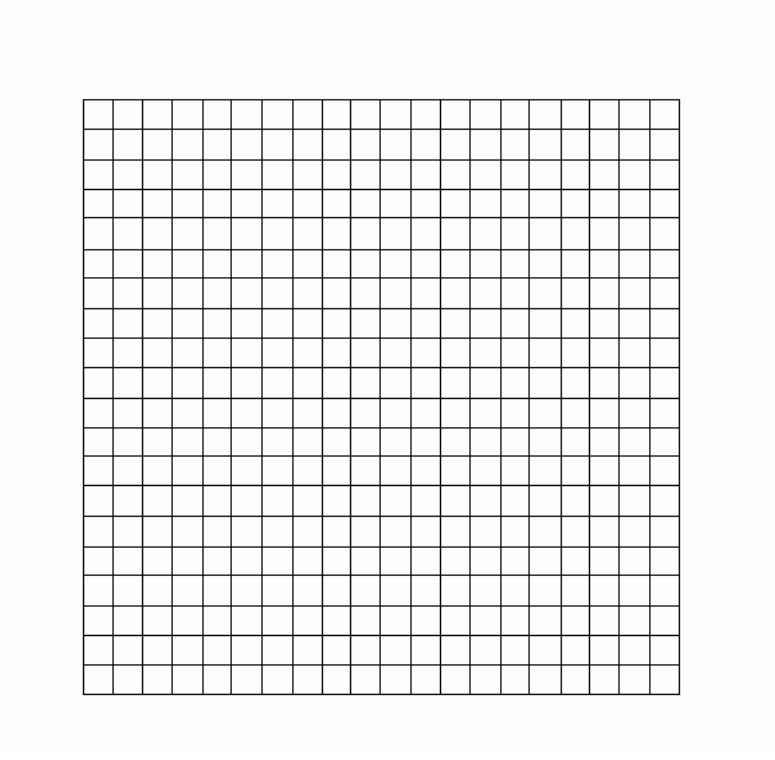
Abhilfe durch *Justieren des Startpunktes*:
Umrechnung in Gerätekoordinaten, runden, Zurückkehren in Weltkoordinaten.

```
100 200
   transform
   round exch round exch
   itransform
moveto
300 0 rlineto stroke
showpage
```

Beispiel: Erzeugung eines Gitters mit exakt gleichen Linienstärken

```
0.5 setlinewidth
100 100 translate
/Pixelgenau
   {
   transform
   round exch round exch
   itransform
   } def
0 1 20
   {
   dup 20 mul 0 Pixelgenau moveto
   0 400 rlineto stroke
   } for
0 1 20
   {
   dup 20 mul 0 exch Pixelgenau moveto
   400 0 rlineto stroke
   } for
showpage
```

Ergebnis:



Einbindung von Rastergrafiken in PostScript

Direkte Spezifikation eines Rasterbildes mit dem Befehl **`image`**:
Dieser Befehl interpretiert die Elemente eines Hexadezimalzahl-Strings als Bitmuster, die die Grauwerte einzelner Pixel repräsentieren.
Das Rasterbild wird in ein Quadrat der Seitenlänge 1 abgelegt, dessen linke untere Ecke im Ursprung liegt. Vor dem Aufruf der Funktion sollte daher der Nullpunkt an die gewünschte Stelle verschoben und das Koordinatensystem skaliert werden.

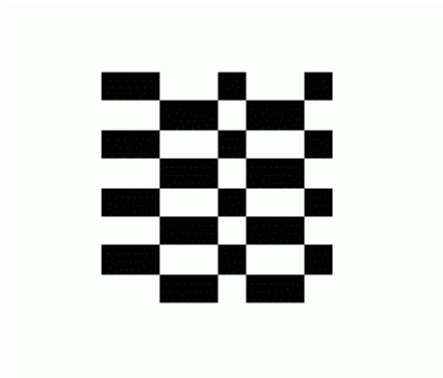
Aufruf-Syntax:
*Spaltenzahl Zeilenzahl Bits_per_Pixel Transformationsmatrix Prozedur* **`image`** →

Darin sind Spalten- und Zeilenzahl die Ausmaße des Rasterbildes. Bits_per_Pixel gibt die Anzahl der Bits an, die je Rasterpunkt zur Verfügung stehen (für den Grauwert); zulässige Parameter: 1, 2, 4 oder 8 (d.h. 2, 4, 16 oder 256 Graustufen).
Transformationsmatrix: für ein Rasterbild, das aus *n* Spalten und *m* Zeilen besteht, sollte diese zur Abbildung auf das Einheitsquadrat [ *n* 0 0 *m* 0 0 ] lauten.
Die Prozedur liefert den Datenstring für das Rasterbild, im einfachsten Fall kann es sich um eine Hexadezimalzahl (in spitzen Klammern) handeln. Interpretation zeilenweise von links unten nach rechts oben. Falls der String nicht genug Daten für das komplette Bild enthält, wird die Prozedur wiederholt so oft aufgerufen, bis das Bild komplett ist. Der Image-Befehl ignoriert überflüssige Bits am Ende der Rasterzeile.

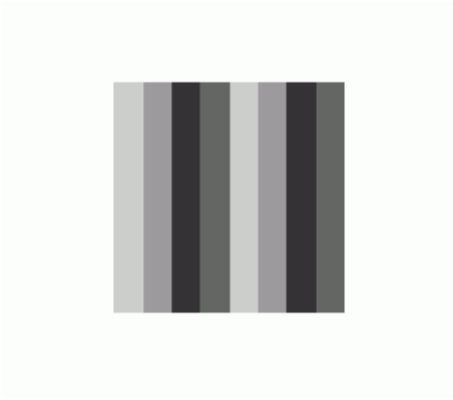
Beispiel:
Quadratisches Rasterbild mit einer Seitenlänge von 1 Zoll

```
300 400 translate
72 72 scale
8 8 1 [ 8 0 0 8 0 0 ] { <C936> } image
showpage
```

Ergebnis:



Hier wird die Hexadezimalzahl C936 = 1100 1001 0011 0110 als Abfolge einzelner Schwarz (0)- und Weiss (1)-Werte interpretiert. Der String codiert 2 Zeilen und wird dann wiederholt.
Variante mit 8 8 4 anstatt 8 8 1 als Argumenten von **`image`**:
Interpretation als Grauwerte 12 (= C), 9, 3 und 6.
Ergebnis:

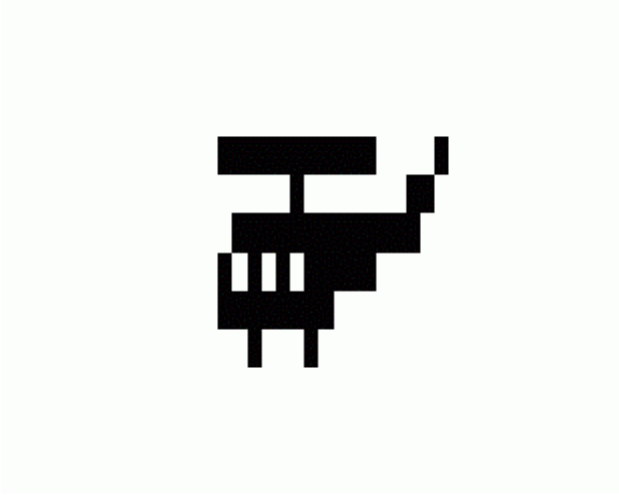


Bei nicht-quadratischem Rasterbild kann die bei Abbildung ins Einheitsquadrat entstehende Verzerrung durch entsprechendes Skalieren ausgeglichen werden.

Icon eines Helikopters, ohne Verzerrungsausgleich:

```
/Helikopter
< DD FF 00 FF 54 1F 80 03 FB F9 00 1E > def

300 400 translate
72 72 scale
16 6 1 [ 16 0 0 6 0 0 ] { Helikopter } image
showpage
```



mit Ausgleich der Verzerrung (verwende **`72 27 scale`** anstatt **`72 72 scale`**):

(dieser Ausgleich kann auch in der Transformationsmatrix vorgenommen werden).