Sind Fonts urheberrechtlich geschützte Computerprogramme?

von Ulrich Stiehl, Heidelberg, 17. November 2005

In der schon zuvor veröffentlichten Dokumentation „Sind Fonts urheberrechtlich geschützte **Kunstwerke**?“ (http://www.sanskritweb.net/forgers/kunstwerk.pdf) wurde ausgeführt, daß der BGH nie einen einzigen der Abertausende von Fonts als urheberrechtlich geschütztes Kunstwerk (§ 2 Abs. 1 Nr. 4 UrhG) anerkannt hat. Ob hingegen Fonts urheberrechtlich geschützte **Computerprogramme** (§ 2 Abs. 1 Nr. 1 i.V.m. § 69a UrhG) sind, hat der BGH noch nicht geprüft, doch gilt aufgrund der hier dargelegten Fakten als sicher, daß der BGH nie auch nur einen einzigen unter den Zehntausenden von Fonts als Computerprogramm anerkennen wird.

Wir beschränken uns hier auf die Erörterung der **TrueType-Fonts**, weil die PostScript-Type-1-Fonts von Adobe seit 1999 nicht mehr produziert werden und bald nur noch von historischer Bedeutung sein dürften.

In den einschlägigen Urheberrechtskommentaren von Fromm/Nordemann, Schricker et al., Rehbinder usw. finden sich nur ganz kurze Bemerkungen zu Fonts. So enthält z.B. der 2004 erschienene neueste Kommentar „Urheberrecht“ von Dreyer/Kotthoff/Meckel nur 6 Zeilen zu Fonts. Dort wird bemerkt (siehe § 69a, Rz. 10), daß Fonts nicht als Computerprogramme geschützt sind, und im übrigen auf diesen Fachaufsatz verwiesen: „Der rechtliche Schutz von Fonts“ von Jaeger/Koglin („Computer und Recht“, Heft 3/2002, Seite 169 ff.). Darin legen die Verfasser dar, daß Fonts keine urheberrechtlich geschützten Computerprogramme darstellen.

1. Die Glyphs

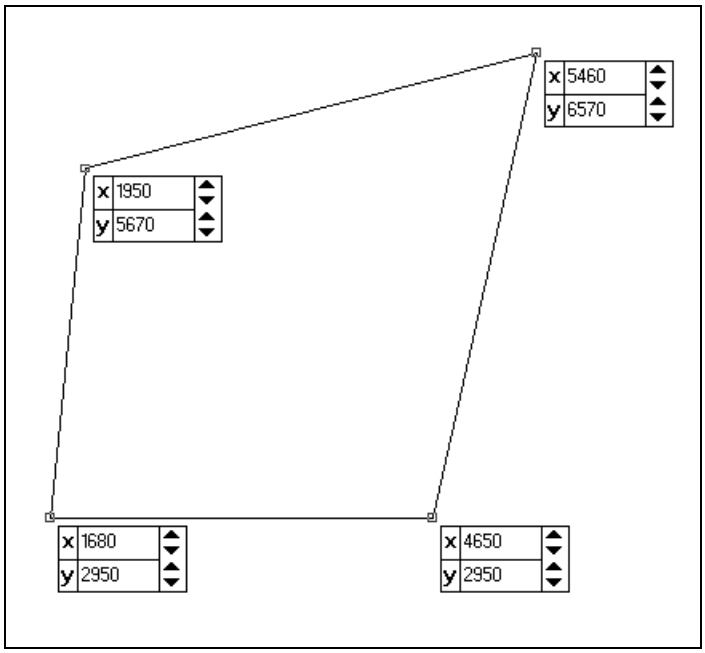
Jaeger/Koglin schreiben in ihrem Fachaufsatz „Der rechtliche Schutz von Fonts“ („CR“, 3/2002, Seite 173):

> Für das Urheberrecht kann die Trennung zwischen Daten und Programmen allerdings nicht aufgegeben werden, da das Urheberrecht gem. §§ 69a ff. nur Programmen besonderen Schutz gewährt. Würde das Computerprogramm mit digitalisierten Daten[50] gleichgesetzt, wäre zum einen der Schutz von Computerprogrammen contra legem begrenzt, da §§ 69a Abs. 1, Abs. 2 S. 1 UrhG das Programm in jeder Gestalt und Ausdrucksform – also auch »analog«, z.B. handgeschrieben auf Papier – schützt. Zum anderen wären klassische Werke, wie Texte, Musik und Bilder dann durch das Digitalisieren stärker als vom Urheberrechtsgesetz vorgesehen geschützt und damit dessen Wertungen, insbesondere die Schutzvoraussetzungen und die Schranken des Urheberrechts, aufgehoben.

Ein Font besteht hauptsächlich aus Glyph-Koordinaten, d.h. x-y-Schnittpunkten von Schriftzeichenkonturen, die als Zahlenreihen gespeichert werden. Wenn man einen TrueType-Font in ein Font-Zeichenprogramm lädt und die Glyphs aller Groß- und Kleinbuchstaben usw. löscht und den Font dann wieder zurückspeichert, schrumpft der Font auf einen winzigen Bruchteil seiner ursprünglichen Dateigröße, also auf eine leere Hülle. Dadurch daß also Glyphs den wesentlichen Hauptteil eines jeden Font ausmachen, steht und fällt die Frage, ob Fonts Computerprogramme sind, mit der Frage, ob die Glyphs Computerprogramme sind oder nicht.

Zwar behaupten Jaeger/Koglin, daß Fonts keine Computerprogramme, sondern „digitalisierte Daten“ sind, doch legen sie für ihre zutreffende Behauptung keinen exakten Beweis vor. Dies holen wir hiermit nach.

Aus mathematischer Sicht sind die Schriftzeichen auf dem Papier „flächige Figuren“. Zur Beschreibung der Kontur und Position einer Figur benennt man die x-y-Koordinaten der Figur in einem Koordinatensystem. Jeder Glyph in einem TrueType-Font beschreibt die Kontur eines Schriftzeichens durch eine Zahlenreihe, welche aus bestimmten x-y-Koordinatenwerten besteht. Betrachten wir dazu das folgende einfache Viereck:



Wenn die obige Figur als Glyph in einem TrueType-Font gespeichert wird, dann wird **nicht die Zeichnung** der Figur in dem Font gespeichert, und es wird auch **nicht der Algorithmus** zum Zeichnen der Figur in dem Font gespeichert, und es wird auch **nicht das Programm** zum Zeichnen der Figur in dem Font gespeichert, sondern es werden nur die obigen Glyph-Koordinatenpunkte als nackte Zahlenreihe oder Zahlenfolge in dem TrueType-Font gespeichert. Wenn wir bei der obigen Figur bei der x-y-Koordinate links oben beginnen (d.h. x = 1950, y = 5670) und im Gegenuhrzeigersinn voranschreiten, würde diese Zahlenreihe wie folgt aussehen:

1950 5670 1680 2950 4650 2950 5460 6570

In Wirklichkeit werden in einem TTF-Font erst alle x-Koordinaten und dann alle y-Koordinaten gespeichert:

1950 1680 4650 5460 5670 2950 2950 6570

Auch diese Form der Zahlen-Speicherung entspricht noch nicht ganz der Wirklichkeit, weil die Zahlen meist durch Zahlenpackverfahren komprimiert werden, um sie platzsparend in der TrueType-Datei zu speichern. Bei Ausnutzung aller Komprimierungen (nicht alle TTF-Fonts machen davon Gebrauch) würde der obige Glyph in einer TrueType-Font-Datei als die folgende digitalisierte hexadezimale Zahlenreihe gespeichert:

13 03 21 13 C3 1B 01 29 51

Aus urheberrechtlicher Sicht ist es gleichgültig, ob in einem TrueType-Font dezimale oder hexadezimale oder gepackte oder ungepackte Zahlenreihen gespeichert werden, denn Zahlenreihen bleiben Zahlenreihen. Entscheidend ist die Erkenntnis, daß ein TrueType-Font weder digitalisierte Zeichnungen der Schriftzeichen noch Programme zur Darstellung dieser Schriftzeichen, sondern lediglich digitalisierte Zahlenreihen enthält, die vom Betriebssystem (z.B. Windows) als x-y-Koordinatenwerte von Schriftzeichen interpretiert werden, und zwar bei Windows durch das GDI (Graphical Device Interface), das anhand der x-y-Koordinatenwerte des jeweiligen Schriftzeichens dessen Kontur am Bildschirm zeichnet, d.h. als ein Schriftzeichen anzeigt. Das Windows-GDI ist ein **Programm**, aber die x-y-Koordinaten der Zeichen im Font sind **kein Programm**, sondern nur Zahlenreihen: Der Font enthält die x-y-Koordinatenzahlen als **Daten**, und Windows enthält das GDI als **Programm**. Die Schriftfirmen verkaufen aber keine GDI-Programme, sondern nur Font-Dateien.

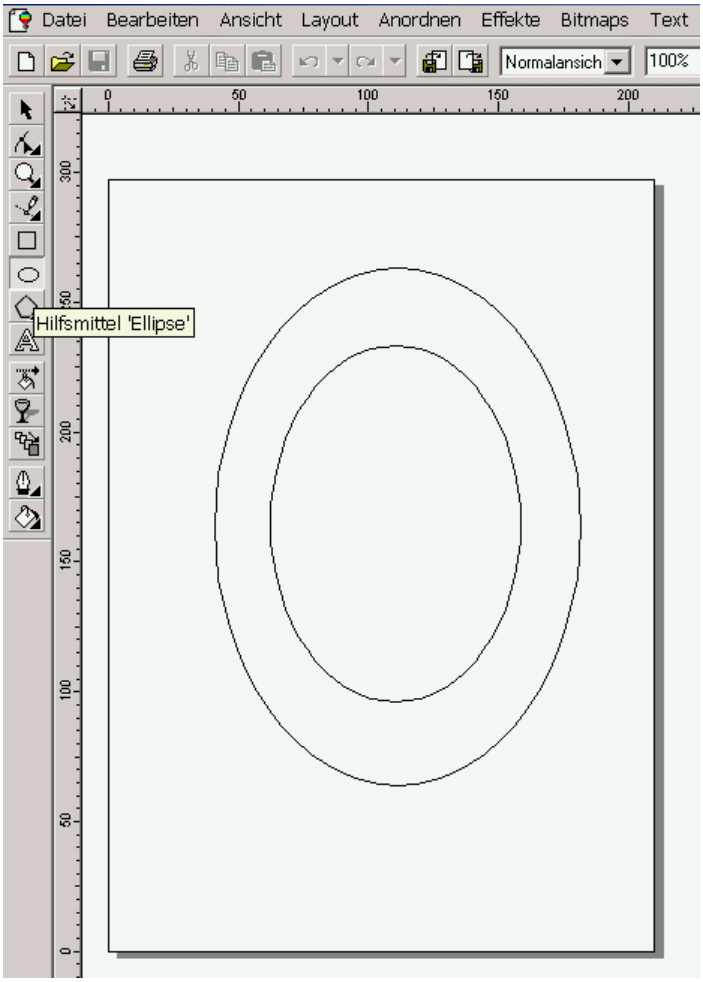
Jeder geistig gesunde Mensch weiß, daß Zahlenreihen (z.B. „3 4 5 6 7“) keine Computerprogramme sind. Wenn daher ein Richter in einem Urteil behaupten würde, daß TrueType-Fonts, mithin z.B. die Zahlenreihe „3 4 5 6 7“ (die in **jedem** TTF-Font vorkommt) urheberrechtlich geschützte Computerprogramme seien, dann ist dieser Richter entweder geistesgestört, oder er begeht bewußt eine Rechtsbeugung.

1.1. Der Fall „Font-Test“

Richter konstruieren gern **fiktive Rechtsfälle**. Gesetzt den Fall, der BGH-Präsident Prof. Dr. Günter Hirsch würde an einem PC sitzen, auf dem das weitverbreitete Zeichenprogramm „**CorelDraw**“ installiert wäre. Experimenti causa würde der Richter nun den Großbuchstaben „O“ zeichnen (Schritt 1) und die Zeichnung als CDR-Datei sowie als TTF-Datei „Font-Test“ speichern (Schritte 2 und 3, siehe folgende Abbildungen). Der ganze Test dauerte nur wenige Sekunden. **Frage:** Hat der BGH-Präsident soeben als eine eigene geistige Schöpfung ein Computerprogramm als ein Sprachwerk in einer Computerprogrammiersprache geschrieben?

„**Computerprogramme**“ sind in einer Computerprogrammiersprache (Assembler, BASIC, Pascal, C usw.) geschriebene „**Sprachwerke**“ (§ 2 Abs. 1 Nr. 1 i.V.m. § 69a Abs. 4 UrhG), die nur dann urheberrechtlich geschützt sind, wenn der Computerprogrammierer sie als „**eigene geistige Schöpfung**“ (§ 69a Abs. 3 UrhG) geschaffen hat. Die eigene geistige Leistung des BGH-Präsidenten bestand hier darin, daß er durch Zeichnen ein Schriftzeichen erstellte, also eine **Zeichnung** (= **kein** Computerprogramm!), die jedoch, wie der BGH im BGH-Urteil I ZR 21/57 vom 30.05.1958 feststellte, kein urheberrechtlich geschütztes „Kunstwerk“ darstellt.

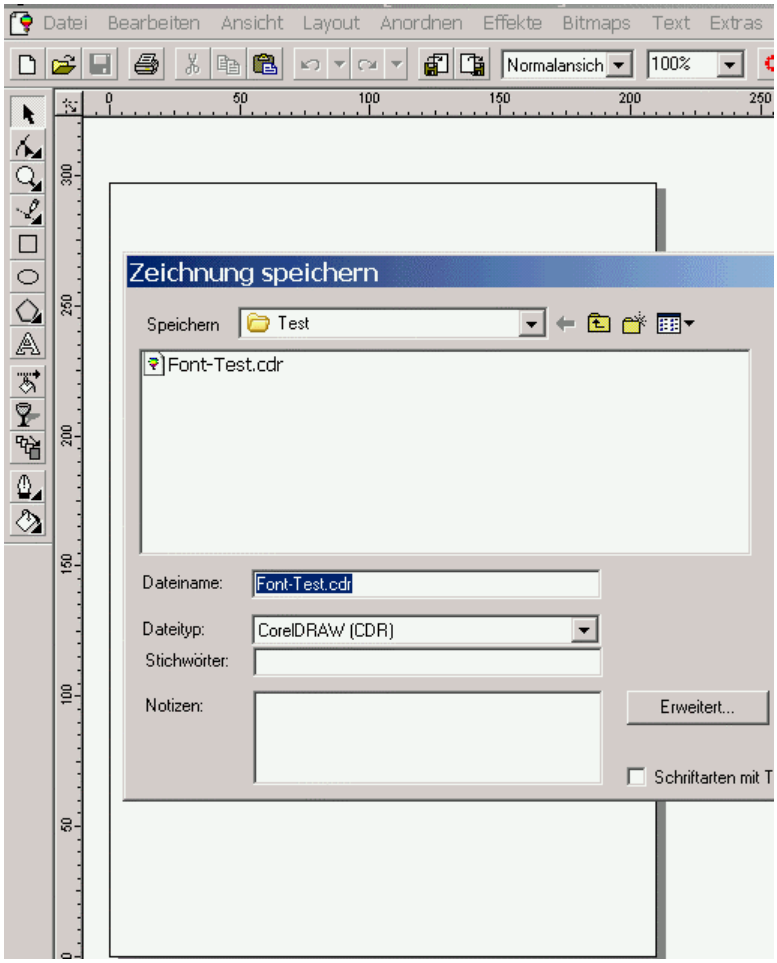
Wer mit einem **Zeichenprogramm** (z.B. „CorelDraw“) eine Bild zeichnet und als CDR-Datei speichert, schreibt kein Computerprogramm, doch das Bild kann ggf. als **Kunstwerk** urheberrechtlich geschützt sein. Das gleiche gilt, wenn ein Font-Designer mit einem Font-Zeichenprogramm, d.h. mit einem Font-Editor („Fontlab“, „Fontographer“ usw.), die Konturen von Schriftzeichen zeichnet und als TTF-Datei abspeichert. Wer also ein Zeichenprogramm oder Font-Editor benutzt, ist kein Programmierer, sondern nur ein Zeichner, d.h. wer Linien und Kurven von Schriftzeichen mit der Maus zieht, programmiert nicht, sondern zeichnet. Eine Zeichnung ist **kein** „Computerprogram“ und **kein** „Sprachwerk“, sondern allenfalls ein „Kunstwerk“, das ggf. urheberrechtlich geschützt ist, sofern eine „**geistige Schöpfung**“ vorliegt, was aber für Buchstaben von Gebrauchsschriften grundsätzlich nicht zutrifft (s. http://www.sanskritweb.net/forgers/kunstwerk.pdf).

Schritt 1: Zeichnen Der Richter klickt mit der Maus auf das Symbol „Hilfsmittel Ellipse“ und zieht dann mit der Maus zwei Ellipsen, die äußere und die innere Kontur des Großbuchstabens „O“.	

Schritt 2: CDR-Datei

Der Richter klickt mit der Maus auf das Menü „Datei“ und wählt dann „Speichern“ als CDR (= CorelDraw) Datei.

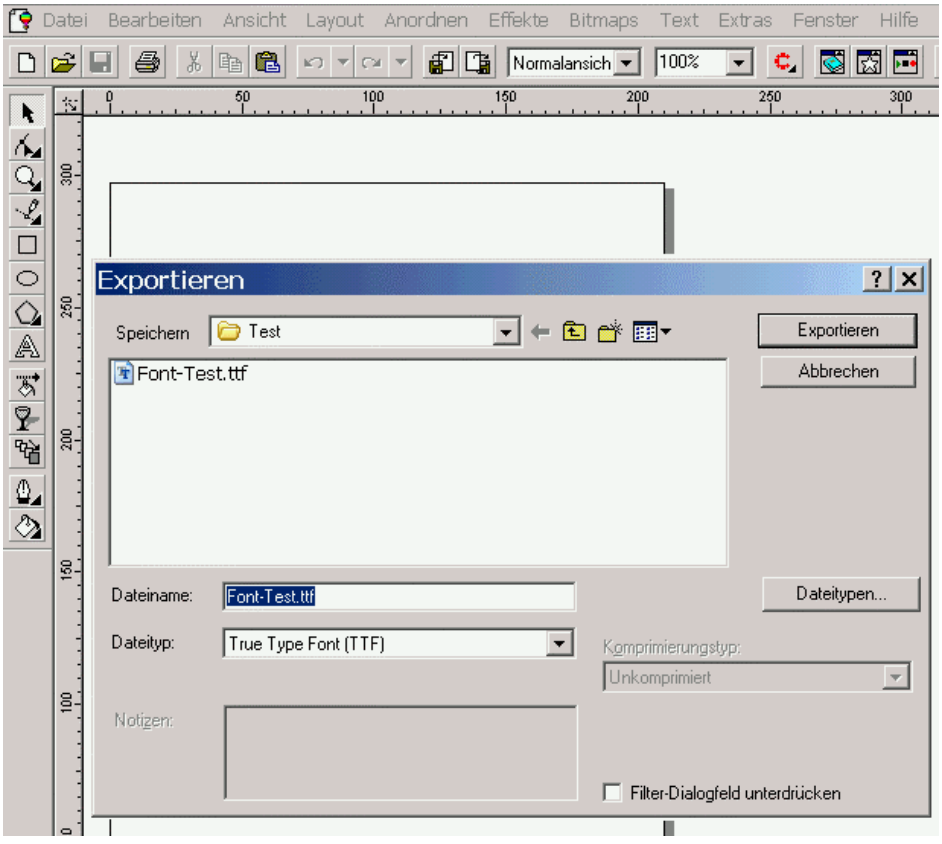
Als Datei-Name wählt er „Font-Test.CDR“.



Schritt 3: TTF-Datei

Der Richter klickt mit der Maus auf das Menü „Datei“ und wählt dann „Exportieren“ als TTF (= TrueType) Datei.

Als Datei-Name wählt er „Font-Test.TTF“.



1.2. Die Datei „Font-Test.ttf“

Als der Richter (s.o. Schritt 3) auf „Exportieren“ klickte und TTF wählte, hat CorelDraw tatsächlich einen TrueType-Font generiert, der hier **ein** Schriftzeichen enthält, d.h. die x-y-Koordinaten des „O“-Buchstabens.

Die folgenden Ausführungen sind höchst technisch, aber unabdingbar, um den **exakten Beweis** zu liefern, daß die Glyphs von TrueType-Fonts **keine Computerprogramme** sind:

Am Anfang der von CorelDraw automatisch erzeugten TrueType-Datei „Font-Test.ttf“ finden wir das Label „glyf“ und dann 4 Bytes danach den Zeiger 000004A4, der auf den Speicherbereich der Glyphs verweist:

```
00000000  00 01 00 00 00 0A 00 80 00 03 00 20 4F 53 2F 32...OS/2
00000010  12 C3 03 CB 00 00 00 AC 00 00 00 4E 63 6D 61 70...cmap
00000020  24 07 67 2C 00 00 00 FC 00 00 03 A6 67 6C 79 66...glyf
00000030  2E 9E 12 22 00 00 04 A4 00 00 01 10 68 65 61 64...head
```

Der Hexadezimalzahlen-Dump 000004A4 bis 0000052B enthält die x-y-Koordinaten des „O“-Buchstabens:

```
000004A0  00 00 00 00 00 02 01 4B 02 02 05 B8 08 4A 00 0B
000004B0  00 17 00 00 01 00 00 01 00 00 01 00 00 01 00 00
000004C0  01 00 00 01 00 00 01 00 00 01 00 00 03 7A FF 60
000004D0  FF 1D 00 00 00 00 00 E3 00 A0 00 A0 00 E4 00 00
000004E0  00 00 FF 1C FF 67 00 EA 01 4D 00 00 00 00 FE B3
000004F0  FF 16 FF 16 FE B4 00 00 00 00 01 4C 07 58 00 00
00000500  FE BB FF 1C FF 1C FE BC 00 00 00 00 01 44 00 E4
00000510  00 E4 01 45 00 F2 00 00 FE 27 FE B5 FE B4 FE 28
00000520  00 00 00 00 01 D8 01 4C 01 4B 01 D9 00 02 01 4B
```

Der entsprechende Dezimalzahlen-Dump 000004A4 bis 0000052B des „O“-Glyph sieht wie folgt aus:

```
000004A0  0 0 0 0 0 2 1 75 2 2 5 184 8 74 0 11
000004B0  0 23 0 0 1 0 0 1 0 0 1 0 0 1 0 0
000004C0  1 0 0 1 0 0 1 0 0 1 0 0 3 122 255 96
000004D0  255 29 0 0 0 0 0 227 0 160 0 160 0 228 0 0
000004E0  0 0 255 28 255 103 0 234 1 77 0 0 0 0 254 179
000004F0  255 22 255 22 254 180 0 0 0 0 1 76 7 88 0 0
00000500  254 187 255 28 255 28 254 188 0 0 0 0 1 68 0 228
00000510  0 228 1 69 0 242 0 0 254 39 254 181 254 180 254 40
00000520  0 0 0 0 1 216 1 76 1 75 1 217 0 2 1 75
```

Wir bringen nun eine Struktur in den Hex-Dump. Wir können vier verschiedene Zahlenreihen unterscheiden: **Rot:** Kopfzahlen, **Grün:** Flag-Zahlen, **Blau:** x-Koordinatenzahlen, **Rosa:** y-Koordinatenzahlen:

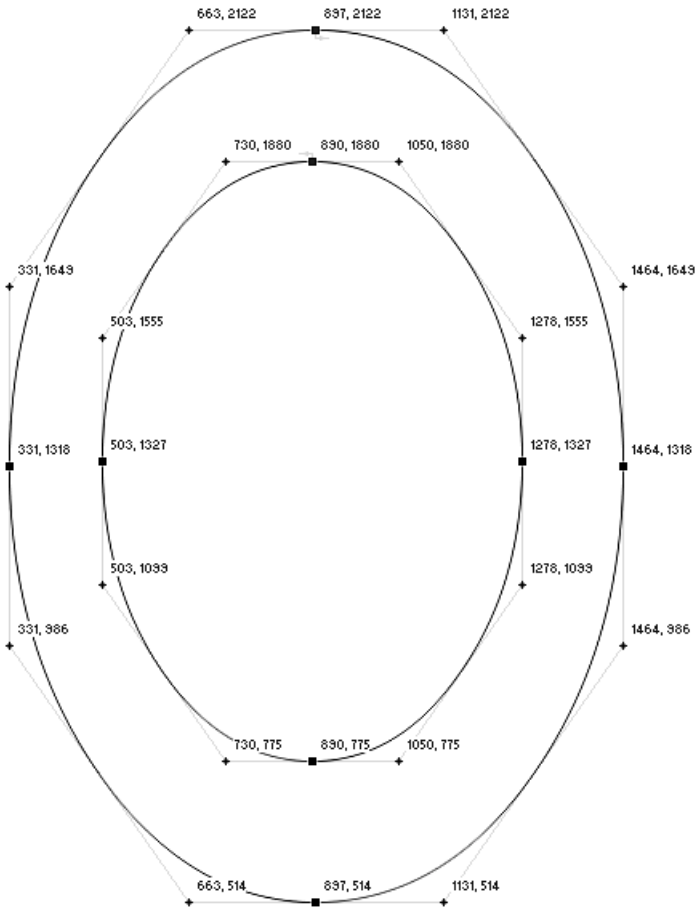
```
000004A0  00 00 00 00[00 02 01 4B 02 02 05 B8 08 4A 00 0B
000004B0  00 17 00 00|01 00 00 01 00 00 01 00 00 01 00 00
000004C0  01 00 00 01 00 00 01 00 00 01 00 00|03 7A FF 60
000004D0  FF 1D 00 00 00 00 00 E3 00 A0 00 A0 00 E4 00 00
000004E0  00 00 FF 1C FF 67 00 EA 01 4D 00 00 00 00 FE B3
000004F0  FF 16 FF 16 FE B4 00 00 00 00 01 4C|07 58 00 00
00000500  FE BB FF 1C FF 1C FE BC 00 00 00 00 01 44 00 E4
00000510  00 E4 01 45 00 F2 00 00 FE 27 FE B5 FE B4 FE 28
00000520  00 00 00 00 01 D8 01 4C 01 4B 01 D9]00 02 01 4B
```

Von dem strukturierten Hex-Dump erzeugen wir einen Dezimal-Dump mit vier verschiedenen Zahlenreihen: **Rot:** Kopfzahlen, **Grün:** Flag-Zahlen, **Blau:** x-Koordinatenzahlen, **Rosa:** y-Koordinatenzahlen:

```
000004A0  0 0 0 0 [ 2 331 514 1464 2122 11
000004B0  23 0 | 1 0 0 1 0 0 1 0 0 1 0 0
000004C0  1 0 0 1 0 0 1 0 0 1 0 0 | 890 -160
000004D0  -227 0 0 227 160 160 228 0
000004E0  0 -228 -153 234 333 0 0 -333
000004F0  -234 -234 -332 0 0 332 | 1880 0
00000500  -325 -228 -228 -324 0 0 324 228
00000510  228 325 242 0 -473 -331 -332 -472
00000520  0 0 472 332 331 473 ] 0 2 1 75
```

Nun erklären wir die vier Zahlenreihen im Detail. Zum besseren Verständnis ist es unbedingt erforderlich, daß man ständig die folgende vermaßte Zeichnung mit den x-y-Koordinatenzahlen des O-Glyph konsultiert. Wir sehen hier die äußere Kontur (Ellipse) sowie die innere Kontur (Ellipse) des O-Schriftzeichens mit allen x-y-Koordinatenzahlen als Dezimalzahlen, die jeweils ganz oben bei der äußeren/inneren Kontur beginnen. Die x-y-Koordinatenzahlen der **inneren** Kontur beginnen oben bei x = 890, y = 1880 und dann weiter im Gegenuhrzeigersinn. Die x-y-Koordinatenzahlen der **äußeren** Kontur beginnen oben bei x = 897, y = 2122 und dann weiter im Uhrzeigersinn. Ferner können wir sehen, daß sich die x-y-Koordinatenpunkte teilweise (bzw. hier beim O-Zeichen im Wechsel) **auf der Kontur** und dann wieder **außerhalb der Kontur** befinden.

Wer bereits von „Bezier-Kurven“ oder von „Splines“ gehört hat, sei darauf hingewiesen, daß ein TTF-Font **weder Algorithmen** zum Zeichnen solcher Kurven **noch Programme** zum Zeichnen dieser Kurven enthält. Natürlich enthält ein TrueType-Font auch keine Buchstabenzeichnungen, sondern nur nackte Zahlenreihen. Mithin sind TTF-Glyphs **weder** „Kunstwerke“ **noch** „Computerprogramme“ im Sinne des Urhebergesetzes.



a) Kopfzahlen: `2 331 514 1464 2122 11 23 0`

- **„2“** bedeutet, daß der Glyph des „O“ aus 2 Konturen, nämlich der äußeren und der inneren Kontur, besteht. (Man vergleiche dazu das Viereck oben auf Seite 2, das nur aus 1 Kontur besteht).
- **„331“** ist die kleinste x-Koordinatenzahl, **„514“** ist die kleinste y-Koordinantenzahl bei dem „O“-Glyph.
- **„1464“** ist die größte x-Koordinatenzahl, **„2122“** ist die größte y-Koordinantenzahl bei dem „O“-Glyph.
- **„11“** bedeutet, daß die elfte Zahl in der x-y-Zahlenreihe die letzte x-Koordinatenzahl ist.
  „23“ bedeutet, daß die dreiundzwanzigste Zahl in der x-y-Zahlenreihe die letzte y-Koordinatenzahl ist.
- **„0“** bedeutet, daß es null Hints gibt.

b) Flag-Zahlen: `1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0`

Die Flags sind Binärzahlen, deren Bits zum Packen der x-y-Koordinatenzahlen verwendet werden **können** (CorelDraw tut dies nicht). Das niedrigste Bit hat eine spezielle Bedeutung: Es enthält den „on“-Wert **„1“**, falls die x-y-Koordinate **auf der Kontur** liegt, und den „off“-Wert **„0“**, falls die x-y-Koordinate **außerhalb der Kontur** liegt. Statt der Bitwerte habe ich unten in die Tabelle die Wörter „on“ bzw. „off“ eingetragen.

c) Zahlenreihen der x-Koordinaten (blau) und y-Koordinaten (rosa)

In der TrueType-Datei wird zuerst die x-Koordinatenzahlenreihe und danach die y-Koordinatenzahlenreihe gespeichert. Die allererste x-Zahl (890) und die allererste y-Zahl (1880) sind **absolute Koordinatenwerte**. Alle nachfolgenden Zahlen sind **relative Koordinatenwerte**. Dies erschwert das Verständnis der Zahlen. Deshalb habe ich eine Tabelle erstellt, die sowohl die relativen als auch die (nur errechneten, aber nicht im TTF-Font gespeicherten) absoluten Zahlenwerte auflistet. Beispiel: Nach der absoluten Ausgangskoordinate x = 890, y = 1880 folgt die relative x-y-Koordinate x = -160, y = 0. Durch Subtraktion 890 – 160 = 730 ergibt sich die neue absolute x-Koordinate. Und durch Addition 1880 + 0 = 1880 ergibt sich die neue (hier unveränderte) absolute y-Koordinate. Man vergleiche die folgende Tabelle mit dem Bild auf der Seite zuvor.

	0	1	2	3	4	5	6	7	8	9	10	11
	on	off	off	on	off	off	on	off	off	on	off	off
X_{rel}	**890**	**-160**	**-227**	**0**	**0**	**227**	**160**	**160**	**228**	**0**	**0**	**-228**
X_{abs}	**890**	**730**	**503**	**503**	**503**	**730**	**890**	**1050**	**1278**	**1278**	**1278**	**1050**
Y_{abs}	**1880**	**1880**	**1555**	**1327**	**1099**	**775**	**775**	**775**	**1099**	**1327**	**1555**	**1880**
Y_{rel}	**1880**	**0**	**-325**	**-228**	**-228**	**-324**	**0**	**0**	**324**	**228**	**228**	**325**
	12	13	14	15	16	17	18	19	20	21	22	23
	on	off	off	on	off	off	on	off	off	on	off	off
X_{rel}	**-153**	**234**	**333**	**0**	**0**	**-333**	**-234**	**-234**	**-332**	**0**	**0**	**332**
X_{abs}	**897**	**1131**	**1464**	**1464**	**1464**	**1131**	**897**	**663**	**331**	**331**	**331**	**663**
Y_{abs}	**2122**	**2122**	**1649**	**1318**	**986**	**541**	**541**	**541**	**986**	**1318**	**1649**	**2122**
Y_{rel}	**242**	**0**	**-473**	**-331**	**-332**	**-472**	**0**	**0**	**472**	**332**	**331**	**473**

Die Glyphs in allen TTF-Fonts sind nichts anderes als nackte Zahlenreihen, die x-y-Koordinaten darstellen. Glyphs in TTF-Fonts sind **keine Zeichnungen** von Schriftzeichen (Zahlenreihen sind keine Zeichnungen). Glyphs sind auch **keine Algorithmen** zum Zeichnen von Bezierkurven oder Splines von Schriftzeichen usw. (Zahlenreihen sind keine mathematische Algorithmen). Glyphs sind auch **keine Computerprogramme** zum Zeichnen von Schriftzeichen auf Bildschirm oder Drucker (Zahlenreihen sind keine Computerprogramme). Folglich sind TTF-Glyphs, die nur Zahlenreihen darstellen und den Hauptteil eines jeden TTF-Font bilden, **keine urheberrechtlich geschützten Computerprogramme** gemäß § 2 Abs. 1 Nr. 1 i.V.m. § 69a ff. UrhG. Quod erat demonstrandum.

Weiter oben wurde konstatiert, daß jeder geistig gesunde Mensch weiß, daß Zahlenreihen (z.B. „1 2 3“ usw.) keine Computerprogramme sind, und daß deshalb ein Richter, der in einem Gerichtsurteil TrueType-Fonts und damit Zahlenreihen als Computerprogramme bezeichnen würde, entweder geistesgestört ist oder bewußt eine Rechtsbeugung begeht. Dies bedarf einer Erläuterung: Zahlen und Zahlenreihen sind in der Wirtschaft unverzichtbar. Man denke an Aktien- und Wechselkurse im Bank- und Börsenwesen, an Preislisten im Groß- und Einzelhandel, an Wasserstände in der Meteorologie, an Meßdaten in Technik und Medizin usw. usf. Wenn folglich Zahlenreihen auf dem Umweg über die TrueType-Fonts urheberrechtlich geschützt wären, würde man damit die ganze Wirtschaft lahmlegen, weil die rund 50.000 TrueType-Fonts, die lieferbar sind, beliebige Zahlenreihen enthalten können, deren freie Benutzung für die Wirtschaft nicht mehr zulässig wäre. Zahlen und Zahlenreihen sind aber gemeinfrei, d.h. niemand darf ein Urheberrecht für Zahlen beanspruchen, die für jedermann frei benutzbar bleiben müssen, weil sonst die ganze Wirtschaft behindert werden würde. Aus diesem Grund sind TTF-Fonts auch nicht als „Datenbanken“ (§ 87a UrhG) urheberrechtlich schutzfähig, zumal auch die übrigen Voraussetzungen von § 87a UrhG („Zugänglichkeit“ der Zahlenreihen in TTF-Fonts, „wesentliche Investition“ für die „Beschaffung oder Darstellung“ der Zahlenreihen usw.) nicht erfüllt wären.

In dem erstinstanzlichen Urteil 28 O 133/97 der 28. Zivilkammer des Kölner Landgerichts wurde behauptet: *„Dabei ist davon auszugehen, daß die 19 streitgegenständlichen Computer-Schriften jedenfalls hinsichtlich der ihnen zugrunde liegenden Computer-Programme Urheberschutz gemäß §§ 69a ff UrhG genießen.“* Mutmaßlich liegt diesem Urteil richterlicherseits eine Geistesstörung zugrunde, die zur Verwechslung der Zeichenprogramme (z.B. „Fontlab“) mit den Fonts (z.B. „F DR NO B“) führte (s. „CR“ 3/2002, Seite 174).

Abschließend sei bemerkt, daß die TrueType-Fonts neben den Zahlenreihen der x-y-Koordinaten der Glyphs, die den wesentlichen Hauptteil eines jeden TTF-Font ausmachen, noch einige andere Zahlenreihen enthalten, z.B. Unicode-Zahlenreihen, die selbstverständlich ebenfalls keine Computerprogramme sind.

2. Die Hints

Jaeger/Koglin schreiben in ihrem Fachaufsatz „Der rechtliche Schutz von Fonts“ („CR“, 3/2002, Seite 173):

> Interessant wird es aber, wenn durch das Hinting besondere Steuerungselemente i.S.d. § 69a UrhG in den Font einbezogen werden. Soweit sie durch das Fonterstellungsprogramm automatisch in die Font-Datei eingefügt werden, fehlt es am Merkmal der »persönlichen Schöpfung«,[52] da nicht ein Mensch, sondern das Programm »Urheber« des Hints ist.[53] Wenn die Hints hingegen individuell eingefügt werden, ist zu unterscheiden: »Klickt« der Schriftdesigner lediglich auf der Abbildung des Buchstabens bestimmte Bereiche an, die bei einer Miniaturdarstellung über- oder unterproportional abgebildet werden sollen, leistet er keine eigene Programmier-, sondern nur Designarbeit. Auch hier setzt das Fonterstellungsprogramm die Steuerungsbefehle in eine computerverständliche Sprache um. Übernimmt der Schriftdesigner hingegen die Umsetzung der Steuerungsbefehle, programmiert er selbst. Nur in diesem Fall, der in der Praxis die Ausnahme bildet, liegt die persönliche Schöpfung eines Computerprogramms vor, die urheberrechtlichen Schutz nach §§ 69a ff. UrhG genießt.

Jaeger/Koglin legen für ihre zutreffende Behauptung keinen exakten Beweis vor. Dies holen wir hier nach.

2.1. Autohinting

Wir haben oben dargelegt, daß die **Zahlenreihen**, die den Hauptteil eines jeden TTF-Font ausmachen, keine Computerprogramme sind, und folglich die Aussage „TrueType-Fonts sind Computerprogramme“ falsch ist. Es stellt sich nur noch die Frage, ob die oft in TrueType-Fonts enthaltenen **Hints** Computerprogramme sind. Wenn Hints urheberrechtlich geschützte Computerprogramme wären, dann wären allerdings **nur diese Hints** geschützt und nicht etwa der ganze Font, der hauptsächlich aus nicht-schutzfähigen Zahlenreihen besteht. Hinzu kommt, daß man mit Font-Editoren (z.B. „Fontlab“) TrueType-Fonts **ohne deren Hints** einlesen und danach völlig neu automatisch „hinten“ und wieder speichern kann. Insoweit kann es juristisch dahinstehen, ob Hints urheberrechtlich geschützte Programme sind, weil man die Hints automatisch neu generieren kann. Zudem ist die Relevanz der Hints, die vor 20 Jahren wegen der damaligen 300-dpi-Laserdrucker usw. ersonnen wurden, stark zurückgegangen, weil heutige Laserdrucker usw. eine viel höhere Auflösung haben.

Vor 20 Jahren hatte ich vier Bücher über Assembler-Programmierung für die Apple-Computer geschrieben („Apple-Assembler“ usw.). Daher ist mir die Programmierung der Mikroprozessoren 6502 und 68000, die in den 1980er Jahren auf Apple-Computern (Apple II und Macintosh) benutzt wurden, noch heute erinnerlich.

In den Jahren ab 1987 hatte Sampo Kaasila, ein Angestellter der Firma Apple, eine Hint-Computersprache für TrueType-Fonts entwickelt. Technisch gesehen handelt es sich dabei um eine Pseudo-Maschinensprache, die nicht für die realen Prozessoren 6502 und 68000 bestimmt war, sondern für einen Pseudo-Interpreter. Diese Pseudo-Maschinensprache bzw. diese Pseudo-Assemblersprache weist teilweise Ähnlichkeiten mit dem 8-Bit-6502-Prozessor auf (z.B. haben die Opcodes je eine Länge von 1 Byte) sowie teils Ähnlichkeiten mit den 68000-Prozessor auf (z.B. werden die 16- und 32-Bit-Parameter „high-byte first“ gespeichert).

Auf der nachfolgenden Seite habe ich ein Hint-Programm in dieser Hint-Programmiersprache dokumentiert. Zu diesem Zweck habe ich den Objektcode der Schrift „Palatino“ (Pala.ttf, Version 1.40 vom 12.10.2000) analysiert und den Assemblersprache-Quelltext rekonstruiert, der von einem Computerprogrammierer hätte geschrieben werden müssen, **falls** der Quelltext jemals von einem Programmierer geschrieben worden wäre. Denn Sampo Kaasila war ein kompromißloser Entwickler: Mit seiner Hinting-Sprache hat er eine derart komplizierte Programmiersprache entwickelt, daß kein einziger Programmierer bekannt ist, der jemals die Hint-Programme eines **kompletten** TrueType-Font wirklich selbst geschrieben hat, d.h. in der Praxis werden Hint-Programme vollautomatisch generiert und sind demnach „computergenerierte Computerprogramme“. Daher sind sie **keine** „persönliche geistige Schöpfung“ und folglich **keine** urheberrechtlich geschützten Computerprogramme, denn „**nur** persönliche geistige Schöpfungen“ (UrhG § 2 Abs. 2) sind schutzfähig.

Daß Hint-Programme vollautomatisch generiert werden, läßt sich durch einfache Berechnungen nachweisen. Nehmen wir die „Palatino“, die aus vier Schriftschnitten (mager, kursiv, halbfett, kursiv halbfett) besteht. Der magere Schnitt Pala.ttf enthält über 230.000 Bytes automatisch generierter Hint-Befehle. Das sind über 80.000 disassemblierte Programmzeilen. Bei allen 4 Schriftschnitten wären dies ca. 320.000 disassemblierte Hint-Befehl-Programmzeilen. (Zum Vergleich enthält die nachfolgende Seite nur 19 Hint-Programmzeilen.) Aufgrund meiner eigenen Erfahrung als Assembler-Programmierer und Autor von Assembler-Fachbüchern schätze ich, daß 1 Programmierer etwa 8 Jahre benötigen würde, um alle 320.000 Befehle zu programmieren. Das beweist, daß schon aus Kosten- und Zeitgründen die Programmierung von Hints gar nicht möglich ist. Daher haben die diversen Schriftfirmen (Linotype usw.), als sie ihre Fonts vom PostScript-Type-1-Format auf TrueType umstellten, innerhalb kürzester Zeit Zehntausende von Fonts vollautomatisch konvertiert und dabei die Hint-Programme von Zehntausenden von TrueType-Fonts vollautomatisch generiert.

Am 1. Oktober 2005 hat der Adobe-Manager Thomas Phinney das „Autohinting“ aller Fonts eingestanden: „*Adobe's own Type 1 and OpenType CFF fonts are all autohinted, but with our own autohinter.*“ (siehe http://www.typophile.com/node/15436), d.h. **alle** Fonts wurden vollautomatisch gehintet („Autohinting“).

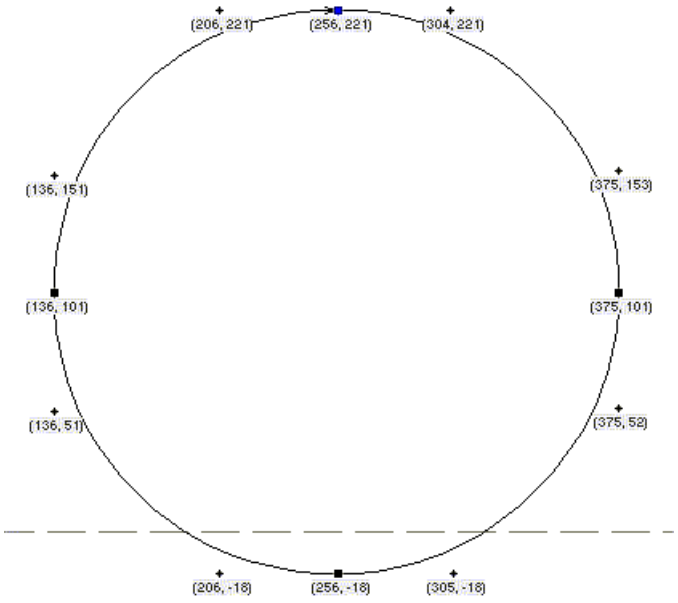
Wenn gelegentlich von „Manual Hinting“ gesprochen wird, wird damit lediglich gemeint, was Jaeger/Koglin als das „Klicken“, d.h. als *„keine eigene Programmier-, sondern nur als Designarbeit“* bezeichnet haben. Mit dem Schreiben von Computerprogrammen in einer Computerprogrammiersprache hat dies nichts zu tun.

Daraus folgt: Hint-Programme in TrueType-Fonts sind zwar Programme im Sinne der Informatik, aber keine Computerprogramme im Sinne des Urheberrechts, weil eine „persönliche geistige Schöpfung“ nicht vorliegt, denn automatisch generierte TrueType-Hint-Programme sind „computergenerierte Computerprogramme“, die nicht urheberrechtlich schutzfähig sind.

Hint-Beispiel: „Satzpunkt“ der „Palatino“ (Pala.ttf, Version 1.40 vom 12.10.2000)

Adresse	**Objektcode** (Maschinensprache)	**Quelltext** (Assemblersprache)
04068A	**B9 0004 FFE0**	PUSHW[2] 4, -32
04068F	**B3 12 1D 36 08**	PUSHB[4] 18, 29, 54, 8
040694	**B8 FFE0**	PUSHW[1] -32
040697	**40 1F 12 1D 36 02 20 12 1D 36 0A 20 12 1D 36 06 00 4A 00 B0 06 09 03 4A 03 AF DF 09 01 00 09 01 09**	NPUSHB [31]: 18, 29, 54, 2, 32, 18, 29, 54, 10, 32, 18, 29, 54, 6, 0, 74, 0, 176, 6, 9, 3, 74, 3, 175, 223, 9, 1, 0, 9, 1, 9
0406B8	**2F**	MDAP[rd]
0406B9	**72**	DELTAP3
0406BA	**5D**	DELTAP1
0406BB	**ED**	MIRP[nrp0,md,rd,1]
0406BC	**2B**	CALL
0406BD	**00**	SVTCA[y-axis]
0406BE	**2F**	MDAP[rd]
0406BF	**ED**	MIRP[nrp0,md,rd,1]
0406C0	**2B**	CALL
0406C1	**31**	IUP[y]
0406C2	**30**	IUP[x]
0406C3	**2B**	CALL
0406C4	**2B**	CALL
0406C5	**2B**	CALL
0406C6	**2B**	CALL

Glyph „Punkt“
am Ende des Satzes
ANSI 046 („period“)



Hexdump der Datei
{Schwarz = Hints}

```
040670  02 05 0C 00 00 17 02 05 01 B6 00 00[00 01 00 88
040680  FF EE 01 77 00 DD 00 0B 00 3D{B9 00 04 FF E0|B3
040690  12 1D 36 08|B8 FF E0|40 1F 12 1D 36 02 20 12 1D
0406A0  36 0A 20 12 1D 36 06 00 4A 00 B0 06 09 03 4A 03
0406B0  AF DF 09 01 00 09 01 09|2F|72|5D|ED|2B|00|2F|ED
0406C0  2B 31 30 2B 2B 2B 2B}25 32 16 15 14 06 23 22 26
0406D0  35 34 36|01 00 30 47 46 31 32 46 46 DD 44 34 31
0406E0  46 45 32 32 46]00 00 01 FF F2 FE C7 01 8D 00 FD
```

2.2. Computergenerierte Computerprogramme

Schon allein die Bezeichnung „**Urheber**recht“ macht deutlich: Ein Urheberrecht ohne Urheber gibt es nicht! So werden z.B. in dem riesigen Katalog „FontBook“, der 25.000 Fonts dokumentiert, viele „Font-Designer“ erwähnt (z.B. „Palatino“: Hermann Zapf), aber der Name eines „Font-Programmierers“ wurde noch niemals in irgendeinem Font-Katalog erwähnt, denn Fonts werden vollautomatisch durch den Computer generiert. Folglich gibt es auch keinen Urheber und folglich gibt es auch kein Urheberwerk. Ohne Urheber kein Werk!

Der Zweck des Urheberrechts ist der Schutz des Urhebers, nicht der Schutz von Computern. In der amtlichen Begründung zu § 1 UrhG steht: *„Die Bestimmung bringt zum Ausdruck, daß der Zweck des Gesetzes der Schutz des Urhebers ist. Nicht das Werk, auf das sich der Schutz bezieht, sondern die Person des Urhebers steht im Vordergrund.“* (s. Marcel Schulze, Materialien zum Urheberrechtsgesetz, Weinheim 1993, S. 83). Dabei gelten als Werke des Urhebers „**nur** persönliche geistige Schöpfungen“ (UrhG § 2 Abs. 2):

- „**Persönlich**“ bedeutet, daß nur eine „Person“, d.h. nur ein **Mensch**, Urheber eines Werks sein kann.
- „**Geistig**“ bedeutet, daß nur der „Geist“, d.h. nur das **menschliche Gehirn**, ein Werk schaffen kann.

Es gibt Rechtssysteme mit animistischen Vorstellungen: Dort glaubt man noch an animistische Geister, z.B. an „Baumgeister“ und „Erdgeister“, und Computer gelten als „Geisteswesen“, und folglich gelten dort auch die von Computern generierten Computerprogramme als „works of the electronic brain“. Das deutsche Recht kennt keinen Animismus. Nur die **im menschlichen Gehirn** entstandeneen Werke gelten als Urheberwerke. „Werke“ von „Elektronengehirnen“ gelten nicht als Urheberwerke. Dabei muß man jedoch unterscheiden:

- **Computerunterstützte Werke** (computer-aided works): Wer ein Textverarbeitungsprogramm benutzt, um ein Buch zu schreiben, oder wer einen Programm-Editor benutzt, um den Quelltext eines Computerprogramms zu schreiben, benutzt den Computer genauso als Hilfe, wie man früher eine Schreibmaschine als Schreibhilfe benutzt hat. Die mit derartiger Unterstützung entstandenen Werke sind Urheberwerke.
- **Computergenerierte „Werke“** (computer-generated „works“): Wer ein Übersetzungsprogramm benutzt, um anhand eines deutschen Textes automatisch eine polnische Übersetzung zu generieren, oder wer einen Font-Editor („Fontlab“, „Fontographer“ usw.) benutzt, um Zehntausende von Hint-Befehlen automatisch zu generieren, benutzt den Computer als Generator von „Werken“, die keine Urheberwerke sind, weil sie nicht von einem menschlichen Gehirn, sondern von dem Computer automatisch generiert worden sind. Wer die Systran-Website http://www.systranbox.com aufsucht, die automatische Übersetzungen anbietet, und einen deutschen Text eingibt und dann auf „Russisch“ oder „Japanisch“ klickt, erhält automatisch eine russische oder japanische Übersetzung generiert, auch wenn er selbst geistig vollständig unfähig ist, russische und japanische Texte zu lesen und zu schreiben. Wer aber geistig unfähig ist, eine Übersetzung zu schreiben, wird nicht dadurch zu deren Urheber, daß er die Übersetzung automatisch generieren läßt.

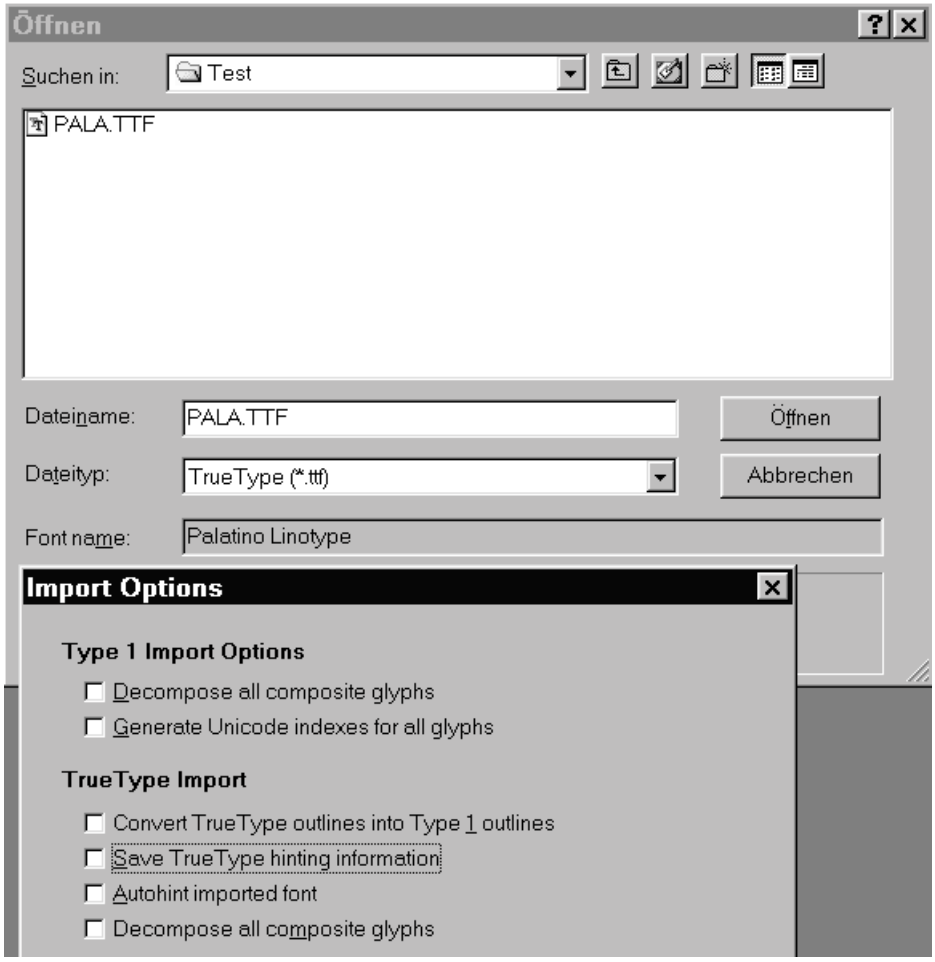
Fiktiver Rechtsfall: Gesetzt den Fall, BGH-Präsident Prof. Dr. Günter Hirsch würde an einem PC sitzen, auf dem der Font-Editor „Fontlab“ installiert wäre, der so konfiguriert sei, daß er beim Laden keine Hints importiert, aber beim Speichern die Hints neu generiert. Nun würde der Richter auf „Pala.ttf“ klicken und damit den TrueType-Font „Palatino“ ohne die Hints importieren. Dann würde er auf „Generate Font“ klicken und den Font mit den neu generierten Hints speichern (siehe Schritte 1 und 2 auf der nachfolgenden Seite). Der ganze Test dauerte nur wenige Sekunden. **Frage:** Hat der BGH-Präsident soeben als eine eigene geistige Schöpfung ein Computerprogramm als ein Sprachwerk in einer Computerprogrammiersprache geschrieben?

Antwort: Computerprogramme müssen „Sprachwerke“ (UrhG § 2 Abs. 1 Nr. 1) sowie „persönliche geistige Schöpfungen“ (UrhG § 2, Abs. 2) sein, und sind nur dann geschützt, „wenn sie individuelle Werke in dem Sinne darstellen, daß sie das Ergebnis der eigenen geistigen Schöpfung ihres Urhebers sind“ (§ 69a UrhG). Da der BGH-Präsident den von „Fontlab“ automatisch generierten Objektcode von Abertausenden von Hints nie gelesen und nie geschrieben hat, kann er auch nicht der geistige Schöpfer dieser Hint-Programme sein. An der Rechtslage würde sich nichts ändern, wenn der Richter zwar fähig wäre, Programme zu schreiben, aber dennoch die Hint-Programme automatisch generiert, um sich jahrelange Programmierarbeit zu ersparen. Da die Hint-Programme nicht von dem Geist eines Menschen erschaffen, sondern von einem Computer, d.h. einer Maschine, erzeugt wurden, gibt es hier überhaupt keinen Urheber und folglich auch kein Urheberwerk. Da aber auch die x-y-Zahlenreihen der Glyphs in dem generierten Font nicht urheberrechtlich geschützt sind, ist der Font „Palatino“ insgesamt nicht als Computerprogramm geschützt. Das gleiche gilt für alle anderen TrueType-Fonts, die allesamt keine urheberrechtlich geschützten Computerprogramme sind.

Schritt 1: Importieren

Der Richter klickt mit der Maus auf „PALA.TTF“ und importiert damit den TrueType-Font „Palatino“.

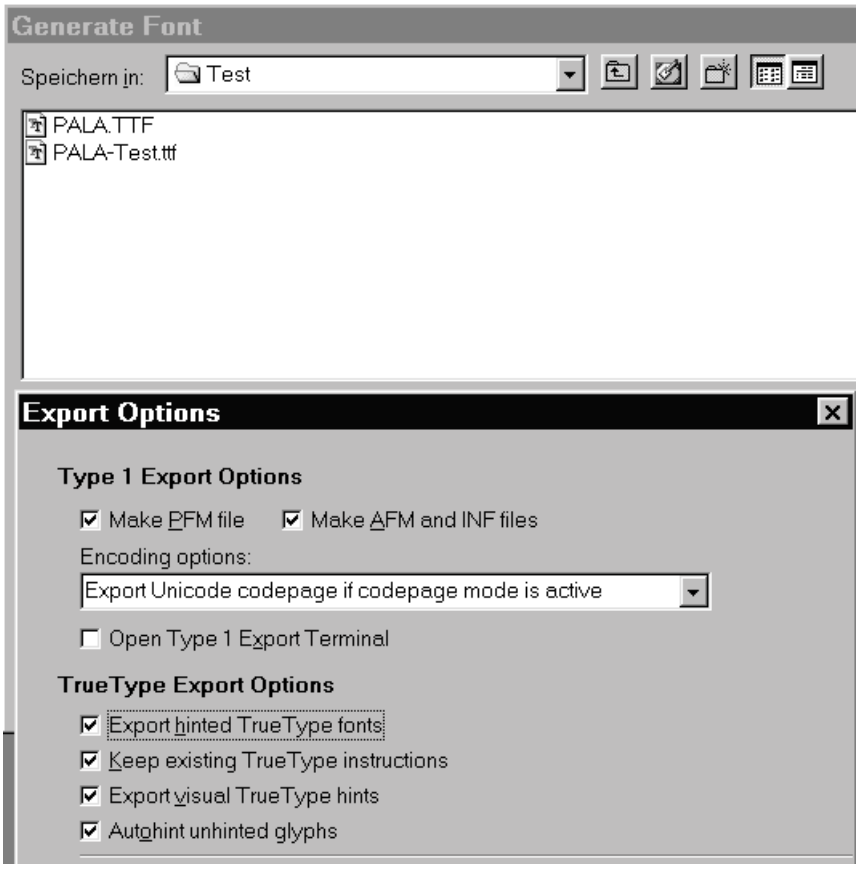
Die Abertausenden von Hints des TrueType-Font werden **nicht** importiert, weil diese Import-Option ausgeschaltet wurde.



Schritt 2: Generieren

Der Richter klickt auf „Generate Font“ und speichert den Font neu unter „PALA-Test.ttf“.

Beim Speichern werden Abertausende von Hints **automatisch generiert**, weil die Option Autohint eingeschaltet wurde.



3. Fazit

Als Fazit steht nunmehr nachweislich fest, daß es keine urheberrechtlich geschützten TrueType-Fonts gibt. Es wäre aber durchaus möglich, urheberrechtlich geschützte Font-Programme zu schreiben, wie das folgende Demo-Programm „FONT.BAS“ zeigt, bei dem fast alle Voraussetzungen des Urhebergesetzes erfüllt sind: Es liegt ein echtes Computerprogramm als Sprachwerk vor, und zwar ein in der Programmiersprache BASIC geschriebener Quelltext, der von einem Menschen als eine persönliche geistige Leistung geschaffen wurde. Allerdings fehlt es bei unserem Demo-Programm an der „Schöpfungshöhe“, d.h. unser Demo-Programm ist derart schlicht und einfach, daß es vom BGH nicht als eine echte „Schöpfung“ anerkannt werden würde. Somit liegt hier nur eine unkreative „geistige **Leistung**“, nicht aber eine kreative „geistige **Schöpfung**“ vor:

Font-Programm	**Bildschirmanzeige**

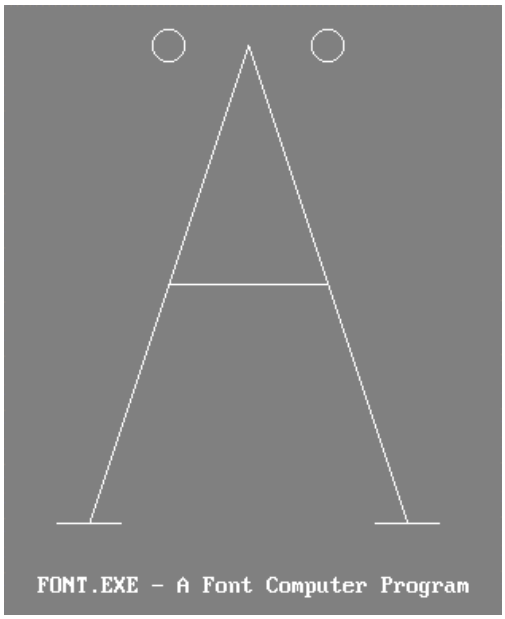
```
REM FONT.BAS
REM Source Code of Demonstration Program
REM "FONT.EXE - A Font Computer Program"
REM Demo written by Ulrich Stiehl, 2005

SCREEN 12: REM 640 x 480

REM Plot the Letter "A dieresis"
REM ============================

LINE (200, 400)-(300, 100): REM Upstroke of A
LINE (300, 100)-(400, 400): REM Downstroke of A
LINE (250, 250)-(350, 250): REM Crossbar of A
LINE (180, 400)-(220, 400): REM Left Foot Serif
LINE (380, 400)-(420, 400): REM Right Foot Serif
CIRCLE (250, 100), 10: REM Left Dieresis Dot
CIRCLE (350, 100), 10: REM Right Dieresis Dot

LOCATE 28, 22
PRINT "FONT.EXE - A Font Computer Program"
END
```



Wenn man das obige Demo-Programm mit den Glyphs der TrueType-Fonts vergleicht, wird sofort deutlich, warum die hauptsächlich aus Glyph-Koordinaten bestehenden TTF-Fonts keine Computerprogramme sind:

- Der **Quelltext** „FONT.BAS“ ist ein in einer Computerprogrammiersprache (BASIC) mit sprachlichen Befehlswörtern (LINE, CIRCLE usw.) geschriebenes **Sprachwerk**. Computerprogramme müssen gemäß Urhebergesetz zwingend Sprachwerke sein (§§ 2 Abs. 1 Nr. 1, 69a Abs. 4 UrhG). Hieran fehlt es bei den TTF-Glyphs, die keine Sprachwerke sind und folglich auch keine Computerprogramme sein können.
- Der **Objektcode** „FONT.EXE“ ist ein **ausführbares Programm**. Wenn man das Font-Programm startet, läuft tatsächlich ein Programm ab, das ein „Ä“ am Bildschirm zeichnet. Hieran fehlt es bei TTF-Glyphs. Ein TTF-Font ist „tot“. Es läuft nichts ab. Ein Font kann von selbst kein einziges Schriftzeichen zeichnen. Ein TTF-Font ist bezüglich der Glyphs nur eine reine Datendatei, die aus nackten Zahlenreihen besteht. Erst das Betriebssystem, das die Glyph-Daten auswertet, zeichnet dann die Schriftzeichen am Bildschirm.

Diese Fakten beweisen, daß TrueType-Fonts **keine** urheberrechtlich geschützten Computerprogramme sind. Welcher geistig gesunde Richter würde jetzt noch erklären, daß TrueType-Fonts Computerprogramme sind? Wenn deshalb ein Richter, der klar bei Verstand ist, sich bewußt über diese evidenten Tatsachen hinwegsetzt und in einem Gerichtsurteil genau das Gegenteil behauptet, dann begeht er die Straftat der Rechtsbeugung. Ein Richter eines unteren Gerichts mag sich zu einer Rechtsbeugung bewegen lassen, nicht aber der BGH. Deshalb steht aufgrund der hier dargelegten Tatsachen als sicher fest, daß der BGB niemals auch nur einen einzigen unter den Zehntausenden von TrueType-Fonts jemals als ein Computerprogramm anerkennen wird.

4. Methoden der Font-Herstellung

In meinen Dokumentationen über die „Font Forging Industry“ (http://www.sanskritweb.net/forgers) habe ich Zehntausende von Fonts untersucht und miteinander verglichen. Die meisten Fonts sind Schriftfälschungen. Zur Herstellung von Fonts werden heutzutage die folgenden Methoden eingesetzt:

1. **Linotype-Methode:** Bei der Linotype-Methode (siehe http://www.sanskritweb.net/forgers/forgers.pdf) werden *bereits vorhandene* Font-Dateien in einen beliebigen Font-Editor („Fontlab“, „Fontographer“, „Type-Designer“, „Font Creator“ usw.) geladen und mit dem Font-Editor die bisherigen Font-Namen und „Copyright“-Vermerke durch neue Font-Namen und „Copyright“-Vermerke ersetzt. Danach klickt der „Designer“ (= Schriftfälscher) auf „Generieren“, und der Font-Editor generiert sodann automatisch „neue“ Fonts. Mit Programmieren hat die Linotype-Methode nichts zu tun. Es gibt Batch-Programme, mit deren Hilfe man bei Abertausenden von Fonts vollautomatisch sämtliche bisherigen Font-Namen und „Copyright“-Vermerke durch andere Font-Namen und andere „Copyright“-Vermerke ersetzen kann. Auf meiner „The Font Forging Industry“-Website (http://www.sanskritweb.net/forgers) wurden von mir Abertausende von Fonts dokumentiert, die nach dieser Linotype-Methode gefälscht worden sind.

2. **Monotype-Methode:** Bei der Monotype-Methode werden nicht nur Namen und „Copyright“-Vermerke von *bereits vorhandenen* Font-Dateien gefälscht (siehe Linotype-Methode), sondern zudem werden auch Änderungen an Glyphs vorgenommen (zu Details siehe http://www.sanskritweb.net/forgers/forgers.pdf). Zum Beispiel lassen sich mit „Fontlab“ in Verbindung mit dem „Python“-Tool vollautomatisch beliebige Änderungen an vorhandenen Font-Dateien vornehmen, z.B. gerundete Serifen durch eckige Serifen usw. auswechseln, magere Fonts in fette Fonts konvertieren, gerade Fonts in schräge Fonts umwandeln usw. Auf diesem Wege werden auch vorhandene TrueType-Hints gelöscht und vollautomatisch neu generiert. Auch lassen sich weitere Font-Varianten (z.B. Shadow-Fonts, Outline-Fonts) automatisch generieren. Mit Programmieren hat die Monotype-Methode nichts zu tun, denn hier schreibt kein Mensch eigene Programme als persönliche geistige Schöpfung, sondern der Computer generiert automatisch die Fonts.

3. **Konverter-Methode:** Im Laufe der letzten 25 Jahre haben sich die Font-Formate wiederholt geändert. Manche Formate sind inzwischen völlig verschwunden (z.B. das Intellifont-Format von Compugraphic), andere Formate werden gerade vom Markt genommen (z.B. das PostScript-Type-1-Format von Adobe). Als z.B. vor über zehn Jahren das TrueType-Format auf den Markt kam, haben die Schriftfirmen mittels Konvertern (= Konvertierungsprogrammen) automatisch alle *bereits vorhandenen* PostScript-Fonts in TrueType-Fonts umgewandelt. Auch mit Font-Editoren in Verbindung mit Batch-Programmen lassen sich vollautomatisch in wenigen Minuten Tausende von Fonts von dem einen in das andere Fontformat umwandeln. Mit Programmieren hat Konvertieren nichts zu tun, d.h. der Benutzer programmiert nicht. Vielmehr erzeugt oder generiert der Konverter vollautomatisch aus dem Fontformat A das Fontformat B. Auf diesem Wege werden auch Windows-Fonts in Macintosh-Fonts konvertiert (z.B. mit „TransType“).

4. **Autotrace-Methode:** Bei der Autotrace-Methode werden Schriftmuster (Ausdrucke der Schriftzeichen) *bereits vorhandener* Fonts mit einem Scanner eingelesen und dann von einem Autotrace-Programm (z.B. „Ikarus“, „ScanFont“ usw.) vollautomatisch in die x-y-Koordinatenzahlen der Glyphs umgewandelt. Vereinzelt werden aber auch heutzutage noch Schriftzeichen neuer Schriften auf dem Papier gezeichnet und dann nach der Autotrace-Methode vollautomatisch in die Glyph-Koordinatenzahlen umgewandelt. Im Handbuch zu „ScanFont“ heißt es: „*With ScanFont you can create a professional looking technically correct digital font in minutes*“, d.h. anstelle jahrelanger Programmierarbeit kann man mittels Autotrace in kürzester Zeit einen Font generieren. Mit Programmieren hat die Autotrace-Methode nichts zu tun.

5. **Vektorgrafik-Methode:** Bei der Vektorgrafik-Methode werden mittels allgemeiner Zeichenprogramme (z.B. „CorelDraw“, „Adobe Illustrator“ usw.) die Konturen von Schriftzeichen gezeichnet und dann direkt zu Fonts konvertiert (z.B. mit „CorelDraw“, siehe oben), oder die Zeichnungen der Schriftzeichen werden in einen Font-Editor (z.B. „Fontlab“) importiert, mit dessen Hilfe dann automatisch die Fonts generiert werden. Anstelle eines allgemeinen Zeichenprogramms (z.B. „Illustrator“) kann man auch einen Font-Editor (z.B. „Fontographer“) zum Zeichnen benutzen. Der Font-Editor dient dann sowohl als Zeichenprogramm als auch als Font-Generator. Mit Programmieren hat die Vektorgrafik-Methode nichts zu tun, denn der Benutzer eines Zeichenprogramms ist kein Programmierer, sondern ein Zeichner.

http://www.sanskritweb.net/forgers