Dipl.-Ing. (FH) CHRISTIAN GREIM

Probleme bei Vierfarbrasterung: Rosetten, Farbdrift und Moiré

Die Störenfriede der Farbreproduktion sind auch in der PostScript-Rasterung unberechenbar – empirisch gefundene Praxislösungen

Im zweiten Teil seiner Diplomarbeit (Teil 1: DD 21–22, w70) ging der Autor auf die Festlegung von Rasterweite, Rasterwinkel und Punktformen in PostScript ein. Diese Zusammenfassung erklärt leicht verständlich, wie Rasterpunktformen definiert und moiréfreie Vierfarbreproduktionen gerastert werden können.

Zunächst eine ziemlich allgemeine Aussage, die man beim Umgang mit PostScript immer im Hinterkopf behalten sollte: PostScript liefert das Modell einer Seite in Zahlen und Befehlen. Erst das Ausgabegerät versucht die möglichst optimale Realisierung dieses Modells. Das klingt wieder ziemlich abstrakt. Ich möchte es veranschaulichen. Zum Beispiel sieht das Modell einer Linie so aus:

100100 moveto
100100 rlineto
stroke
showpage

Übersetzt würde dies heißen: gehe zum Punkt mit den Koordinaten 100100; ziehe von dort aus eine Linie zum Punkt, der relativ zum ersten Punkt bei den Koordinaten 100100 liegt; ziehe die Linie, zeige die Seite. Hier werden einige fragen, wer gibt in meiner Firma solche Befehle an den Computer? Die wenigsten werden direkt so programmieren. Aber die meisten Programme wie *Pagemaker*, *Corel Draw*, *TypoScript* oder 3B2 erzeugen solche Befehle. Davon merkt der Benutzer nichts. Die zweite Frage, die sich ergibt, lautet: Woher weiß der Rechner, welche Linienstärke und welche Farbe verwendet werden soll? Dazu müssen irgendwo vorher im PostScript-Programmtext die entsprechenden Definitionen stehen. Sie würden zum Beispiel lauten:

0.5 setgray
20 setlinewidth

Übersetzung: setze einen Grauwert von 0,5; setze eine Linienstärke von 20. Die 20 bezieht sich auf die aktuelle Einheit, die aber auch variiert werden kann, was hier zu erklären aber zu weit führen würde. Die letzten beiden Programmschritte sind *Definitionen*, im Gegensatz zu den vorherigen *Befehlen*. Die Befehle werden vom

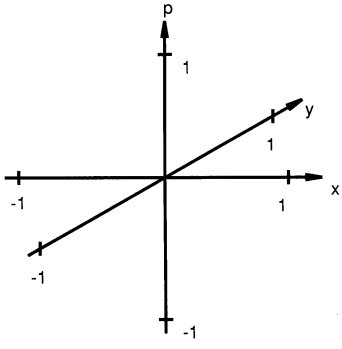

Abbildung 9: Mathematischer Raum der Spotfunktion

Interpreter sofort ausgeführt, während die Definitionen hinterlegt werden, damit spätere Befehle darauf zugreifen können. Aber was ist nun wieder der Interpreter? Der Interpreter ist ein Computerprogramm, das die Aufgabe hat, die in PostScript beschriebene Seite in Nullen und Einsen umzuwandeln, etwa so:

000111000
001000100
001000100
000111000

Natürlich muß man sich diese Folge von Nullen und Einsen viel größer vorstellen. Bei einer Auflösung des Belichters von etwa 1000 Linien pro cm, ergibt sich bei einem DIN A4-Blatt ein Datenfeld von Nullen und Einsen in der Größe von 21000 mal 29700, was wiederum einer Gesamtmenge von 623700000 entspricht. Selbst bei sparsamster Speicherung der einzelnen Bits ergibt sich ein Speicherbedarf von etwa 78 MegaByte.

Die Ausgabegeräte wie Laserdrucker oder Laserbelichter brauchen zur Belichtung aber irgendwann einmal eine solche simple Folge Bits, die einen von zwei Zuständen anzeigen, damit sie entsprechend einen Punkt setzen oder nicht. Beim Laserwriter ist dieser Interpreter ein Computerprogramm, das direkt im Drucker abgearbeitet wird. Beim Laserbelichter ist der Computer, auf dem das Interpreterprogramm läuft, in der Regel der sogenannte RIP (Raster Image Prozessor). Was manchmal zu der seltsamen Kombination führt, daß der RIP eine weitaus höhere Rechenleistung hat, als der eigentliche Computer, an dem DTP gemacht wird. Wie dem auch sei, der RIP braucht eigentlich nur zwei Angaben über das Ausgabegerät, nämlich die Auflösung des Ausgabegerätes und die Größe der Ausgabeseite, eventuell noch eine Gradationskurve des Ausgabegerätes. Da dies nur ziemlich wenige Angaben sind, gibt es auch sogenannte Software-RIPs. Das sind Programme, die auf normalen Computern betrieben werden und an die man die meisten Ausgabegeräte hängen kann. Als Beispiel seien hier die *Hyphen*-

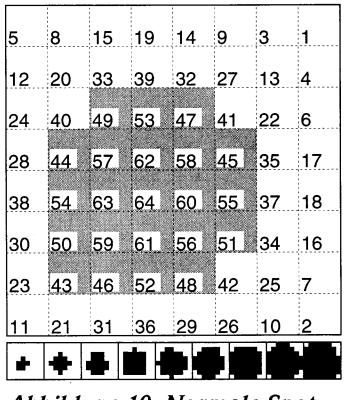

Abbildung 10: Normale Spotfunktion mit runden Punkten in der Auflösung 8x8

RIPs genannt und das Programm *Freedom of Press*. Beide gibt es für mehrere Rechner. Alle Hersteller von PostScript-RIPs in irgendeiner Form müssen Lizenzgebühren an *Adobe* zahlen.

Die bisherigen Fachbegriffe lassen sich leicht aus Computerwörterbüchern entnehmen, die PostScript-Grundlagen sind meiner Meinung nach am besten in *Das große Buch zu PostScript* von TOBIAS WELTNER beschrieben. Nun geht es um einige Begriffe, über die sich die Fachwelt nicht einig ist. Es geht um Punkte, wobei Sie lediglich die prüfungsmäßige Bedeutung beiseite legen dürfen. Ein Bild besteht aus Punkten – logisch!

Aber sind nun die Rasterpunkte im Druck, die einzelnen Graustufen, oder die Punkte des Ausgabegerätes gemeint? Die Rasterpunkte will ich wie gewohnt auch als solche bezeichnen.

Schwieriger ist es da schon, die kleinstmöglichen Punkte zu benennen, die beispielsweise ein Belichter erzeugt. Belichterpunkte? Aber was wäre da mit Punkten aus einem Laserdrucker? Häufig wird zur Angabe der Ausgabeauflösung die Bezeichnung dpi (dots per inch) angegeben. Dies gilt sowohl für Laserbelichter, als auch für Laserdrucker. Deshalb halte ich die Bezeichnung Dot für sehr passend, da es auch ein bißchen lautmalerisch auf einen kleinen Farbklecks hinweist.

Am schwierigsten wird die Sache mit der letzten Art von Punkten, die ich im weiteren Verlauf noch brauche. In PostScript geht es nur um das abstrakte Modell einer Seite. Also stellt man sich ein Bild aus vielen kleinen Quadraten (Vorsicht, es können auch Rechtecke sein!) aufgebaut vor. Jedes dieser »Quadrate« hat einen bestimmten Grauwert, in PostScript zwischen 0 und 255, wobei 0 Schwarz repräsentiert. Diese »Quadrate« haben wirklich überhaupt nichts mit den Rasterpunkten zu tun! Aus diesen rein gedachten »Quadraten« besteht das Bild in PostScript zunächst, deshalb möchte ich sie Pixel (Picture-Elements), also Bildelemente nennen. Das ist zwar wieder Englisch, aber an CMYK als Farbbezeichnungen stört sich ja auch keiner mehr. Also: Ein Bild besteht zunächst aus *Pixeln*, es wird später gerastert in *Rasterpunkte* und jeder Rasterpunkt besteht wieder aus mehreren *Dots*.

(Lesen Sie weiter auf w17)

5	7	15	28	38	49	52	63	51	46	39	27	16	9	3	1
13	20	33	54	76	85	106	109	105	87	78	58	37	25	11	4
24	44	64	91	115	135	138	145	141	131	118	92	65	43	22	6
31	69	102	126	148	166	170	187	172	165	151	124	104	70	30	14
53	94	122	157	183	195	202	209	200	194	179	158	123	93	61	29
77	117	154	181	199	218	224	229	227	215	197	178	155	112	72	40
86	134	163	190	219	232	237	244	240	234	216	191	160	130	83	48
98	137	175	203	225	242	249	254	248	236	222	207	171	142	99	60
111	144	186	211	230	247	255	256	252	245	231	210	180	146	110	62
103	140	174	205	221	241	250	253	251	239	223	201	169	143	96	59
80	128	161	193	217	233	238	246	243	235	214	192	167	132	84	47
74	119	152	176	198	212	220	228	226	213	196	185	153	113	79	41
50	88	127	159	182	189	206	208	204	188	177	156	121	89	57	26
35	66	101	120	150	164	168	184	173	162	149	125	107	68	34	17
18	45	67	90	114	133	136	147	139	129	116	95	71	42	23	8
10	19	36	55	75	82	97	108	100	81	73	56	32	21	12	2

Abbildung 11: Wie Abb. 10, aber in der Auflösung 16 x 16

(Fortsetzung von w12)

Eine Frage bleibt bei unserem Linienbeispiel noch offen, wie nämlich der Grauwert von 0.5 eigentlich dargestellt wird. Wenn in PostScript irgendwelche Definitionen vom Anwender nicht gemacht werden, die für einen bestimmten Befehl gebraucht würden, dann greift der Interpreter einfach auf vom System vorgegebene Werte zurück. Das war auch der Grund, warum der Linienbefehl auch ohne die Definition von linewidth und gray funktioniert hat. Welche Definition hätte der Grauwert denn verlangt? Ein Bündel von Definitionen, das mit dem Befehl setscreen gegeben wird. Man wendet den Befehl etwa so an: Rasterweite Rasterwinkel Punktform *setscreen*. Die Rasterweite muß in Linien pro inch angegeben werden, also entspräche eine Rasterweite von 150 etwa einem 59er Raster, da ein inch 2,54 cm lang ist. Die Winkelangabe wird in Grad gemacht, also zum Beispiel 0,45 oder 60. Nun kommt aber das große Problem, wie man eine Punktform beschreibt, die sich mit dem Grauwert ändert. Erschwerend kommt hinzu, daß man ja vorher nicht weiß aus wieviel Dots eigentlich ein Rasterpunkt besteht, und daß das Ganze noch unabhängig von der Winkelung sein soll. Dazu haben sich die Entwickler von PostScript wieder eine kleine Besonderheit einfallen lassen.

Und hier wird es jetzt sehr abstrakt, wie das eben mathematische Modelle so an sich haben. Dazu denkt man sich eine Rasterzelle, die durch ein dreidimensionales Koordinatensystem repräsentiert wird, wie es in der Abbildung 9 gezeigt ist. Dieses besteht zunächst aus den beiden Ortskoordinaten x und y, womit die Lage des Punktes innerhalb der Grauzelle gemeint ist. Beide Koordinaten müssen zwischen −1 und +1 liegen. Und aus diesen beiden Koordinaten errechnet der Interpreter eine Priorität, die wieder zwischen −1 und +1 liegen kann. Bleibt die Frage, was eine Priorität ist. Zunächst errechnet der Interpreter aus der Rasterweite und aus der Auflösung des Ausgabegerätes, aus wievielen Dots ein Rasterpunkt besteht. Der Interpreter steht nämlich vor dem Problem, daß er eine Fläche mit einem bestimmten Grauwert hat, aber keinen echten Grauwert darstellen kann, sondern nur einen bestimmten Prozentsatz von Dots setzen oder nicht setzen kann. Die Priorität gibt ihm nun an, welche Dots er zuerst setzen soll. Die Dots mit der höchsten Priorität werden zuerst gesetzt. Das hört sich immer noch ziemlich verworren an. Der Interpreter prüft zuerst, aus

wievielen Dots ein Rasterpunkt besteht, zum Beispiel 8 mal 8 Dots. Auf diese 8 mal 8 Punkte muß nun das Modell angewendet werden. Dazu werden erst einmal im Bereich von −1 bis +1 acht gleichabständige Punkte gesucht. Diese Liegen bei −0,875, −0,625, −0,375, −0,125, 0,125, 0,375, 0,625 und 0,875. Die Werte sind deshalb so »krumm«, weil man an den Enden nur eine halbe Schrittweite nehmen darf, in diesem Fall 0,125, während die Punkte untereinander 0,25 voneinander entfernt liegen. Nur so sind die Anschlußstellen zur nächsten Rasterzelle gleichmäßig. Nun sucht also der Interpreter zum Beispiel die Priorität des Punktes an der Stelle mit den Koordinaten −0,625; 0,125. Dabei kommt die Angabe der Punktform ins Spiel, auf die ich schon die ganze Zeit hinaus will. Man gibt dazu dem Interpreter eine Funktion an. Allerdings keine Funktion wie üblich in der Form, man gibt ein x an und erhält einen y-Wert, also f(x) = y, sondern in der Form, man gibt die Koordinate mit ihren zwei Werten x und y an und erhält die Priorität p, also f (x, y) = p. Das ist nun leider auch wieder nicht unbedingt eine Funktion im streng mathematischen Sinne, sondern im programmiertechnischen Sinn, das heißt, man definiert einen Programmteil, dem je ein Wert für x und einer für y übergeben werden und dieser Programmteil gibt daraufhin einen Wert p aus, der, wie man nicht vergessen darf, zwischen −1 und +1 liegen sollte. Dieser spezielle Programmteil wird in PostScript allgemein als Spotfunktion bezeichnet. Diese Programmfunktion muß natürlich an allen Stellen definiert sein, wenn man die Druckerauflösung nicht weiß. Die Auflösung kann man aber über ein paar Befehle herausbekommen und mit in die Funktion verarbeiten. Dann kann man sich natürlich auf die wirklich gebrauchten Werte beschränken. Wir sind aber noch nicht ganz am Ende, da man nun erst Werte zwischen −1 und +1 hat. Diese müssen noch in ganz bestimmte Werte umgewandelt werden. In unserem Beispiel hatten wir 8 mal 8 Punkte, also sind 65 Graustufen darstellbar von »alle Dots hell« bis »alle Dots dunkel«. Also verteilt der Interpreter je nach Reihenfolge in der Priorität Werte von 0 bis 64, so daß jedem Punkt einer dieser neuen Prioritätswerte zugeordnet wird. In dieser Reihenfolge werden also die Punkte je nach Grauwert eingeschaltet. Wie man sich das ganze vorstellen kann, zeigt Abbildung 10. Ganz oben steht der Programmierbefehl für die Spotfunktion, in diesem Fall: dup mul exch dup mul add 1 exch sub. Das sind die PostScript-Befehle. Etwas ausführlicher könnte man auch schreiben: Dupliziere multipliziere vertausche dupliziere multipliziere addiere 1 vertausche subtrahiere. Nur was soll da ständig addiert, multipliziert oder sonstetwas werden?

PostScript hat eine Besonderheit, den Stack oder Stapel. Auf diesen Stapel kann man alle möglichen Werte ablegen, aber wirklich wie bei einem Stapel kann man immer nur oben drauf legen und auch oben wieder runternehmen. Alle anderen Methoden bergen gewisse Risiken, da bin ich zum Leidwesen meines Chefs Experte. Wenn diese Spotfunktion begonnen wird, liegen bereits zwei Werte auf dem Stapel, nämlich unten der x- und oben der y-Wert. Nach dem Befehl dupliziere (dup) liegt der oberste Wert einfach doppelt auf dem Stapel. Mit multipliziere (mul) werden die obersten Werte miteinander multipliziert und das Ergebnis wird auf den Stapel gelegt. Wenn man das ganze nachvollzieht, hat das etwas von einem Denkspiel. Da wäre noch die 1. Wenn man einfach nur 1 eingibt, wird einfach die 1 auf den Stapel gelegt. Das ganze brauchen Sie auch nicht genau nachzuvollziehen. Nur das Prinzip sollte klar sein. Darunter sieht man in Abbildung 2 ein Feld, das aus 8 mal 8 Quadraten besteht. Jedes Quadrat repräsentiert einen Dot. In jedem Dot steht eine Zahl zwischen 1 und 64. Diejenigen Dots, die bei einem Tonwert von etwa 30% geschwärzt werden würden, sind mit einem Raster hinterlegt. In der nächsten Zeile sind verschiedene Graustufen dargestellt wie die Dots bei der vorgegebenen Auflösung besetzt wären. In der letzten Zeile werden die Grauwerte einfach grob gerastert, aber in der Auflösung des Ausgabegerätes dargestellt.

Beispiele für unterschiedliche Raster

Leider ist diese ganze Theorie notwendig, um einigermaßen zu verstehen, worum es bei den ganzen Rasterungsproblemen eigentlich geht. Nebenbei haben wir aber im letzten Absatz den Zusammenhang zwischen Belichterauflösung, Rasterweite und Anzahl der möglichen Graustufen kennengelernt. Indem man nämlich die Belichterauflösung durch die Rasterweite teilt, bekommt man heraus, wieviele Dots die Kante der Rasterzelle lang ist, zum Beispiel 5, 9 oder 16. Diese Zahl muß man nur noch quadrieren, und man erhält die Anzahl der möglichen darstellbaren Graustufen. Also kann man mit einer Rasterzellenkantenlänge von sechzehn 256 Graustufen darstellen, genausoviele, wie PostScript (Level 1) verwalten kann. Allerdings ist es manchmal sinnvoll, mehr Graustufen zur Verfügung zu haben, um bestimmte Gradationen verwirklichen zu können. Abbildung 10 zeigt noch einmal die erste Spotfunktion, aber eben in der Auflösung 16 mal 16. Wer sich die einzelnen Zahlen, die die Prioritäten repräsentieren, genau angesehen hat, hat festgestellt, daß niemals ein Punkt dieselbe Priorität wie ein anderer Punkt hat, obwohl das von der Symmetrie dieser Funktion her durchaus möglich wäre. Da ist PostScript wieder einmal ein bißchen schlauer. Wenn man nämlich zwei oder mehr Werte mit gleicher Priorität bekäme, entscheidet PostScript vorher mit Hilfe eines Zufallszahlengenerators, in welcher Reihenfolge diese Dots gesetzt werden sollen. Leider wirkt dieser Zufallszahlengenerator bei manchen Interpretern nur zeilenweise. Aber um Ihnen zu zeigen, daß wirklich fast alle Formen möglich sind, habe ich als etwas exotisches Beispiel ein Dreiecksraster gewählt, das in der Abbildung 12 gezeigt wird. Wer noch exotischere Punktformen und genauere Erklärungen zur Spotfunktion haben will, der sei auf den Vortrag von Dr. KARL HALLER und WOLFRAM FISCHER auf dem Fogra-Symposium vom 17. und 18. 4. 1991 in München verwiesen. Dazu läßt sich über Dr. KARL HALLER an der FH München auch eine DOS-Diskette gegen Gebühr beziehen. Die Abbildungen 10 bis 12 wurden mit einem speziellen PostScript-Programm erzeugt, mit dem man sehr schön verschiedene Spotfunktionen veranschaulichen kann. Die Anwendung dieses Programms wird besonders im nächsten Teil dieser Artikelserie einiges anschaulicher machen. Es wurde von BERND FLACHSBART geschrieben, einem Studenten an der Fachhochschule für Druck in Stuttgart. Ich habe es an manchen Stellen etwas verändert. Es kann über ihn, Breslaustr. 46, W-3302 Cremlingen bezogen werden, oder über mich. Die Sharewaregebühr bei regelmäßiger Benutzung beträgt 50,– DM. Es ist ein PostScript-Text, der mit einem Editor bearbeitet und auf jedem PostScript-Ausgabegerät verarbeitet werden kann. Besonders interessant an den Abbildungen 10 bis 12 sind die Zahlen in den Quadraten.

3	12	20	41	61	81	109	129	103	80	52	34	25	13	4	1
8	18	27	46	68	96	128	147	121	89	71	48	30	17	7	2
11	24	38	53	84	111	139	170	141	106	75	56	36	23	9	5
15	28	49	69	93	123	154	179	150	117	97	72	44	32	16	6
19	40	57	83	108	142	168	192	164	143	107	79	58	39	22	10
31	50	66	95	127	158	176	196	183	153	124	98	65	47	29	14
35	60	77	113	133	163	186	213	189	165	137	112	82	59	33	21
43	70	91	120	156	182	202	219	198	180	151	125	94	67	45	26
55	78	110	138	171	195	214	227	215	194	167	140	105	85	54	37
64	92	122	155	184	199	221	236	216	205	178	157	119	88	63	42
74	101	130	161	187	210	231	239	225	207	190	169	135	104	76	51
90	126	148	181	206	220	234	247	237	223	201	177	152	115	87	62
100	136	159	185	208	226	243	250	242	228	212	193	160	134	102	73
118	146	172	204	217	235	244	254	245	233	222	203	175	149	116	86
132	162	188	209	229	240	249	255	251	241	230	211	191	166	131	99
145	173	197	224	232	248	252	256	253	246	238	218	200	174	144	114

Abbildung 12: Exotische Spotfunktion in Dreiecksform

14	45	104	152	116	47	17	4	7	42	114	154	109	44	19	2
28	82	147	161	126	71	33	15	30	84	143	164	131	78	35	16
73	93	201	209	184	100	79	38	76	96	186	207	196	91	74	50
146	203	236	241	224	182	139	105	134	195	229	244	232	187	140	106
163	210	251	253	249	206	169	149	171	217	238	256	240	220	170	150
144	194	233	237	234	200	130	113	138	190	221	239	231	189	122	102
72	87	192	208	185	95	81	51	77	94	198	216	202	85	83	46
32	67	136	172	125	75	36	11	27	69	137	158	141	61	34	20
10	49	103	155	112	52	12	1	18	48	115	153	111	40	13	3
26	64	135	167	118	68	24	9	31	65	148	162	133	60	29	8
62	89	193	212	191	99	58	39	63	88	197	214	188	98	80	37
142	177	227	243	222	183	124	101	127	204	235	245	228	178	132	107
165	219	246	254	250	205	166	156	168	211	242	255	247	215	160	151
120	176	223	252	226	173	129	110	117	199	225	248	230	175	145	108
66	90	180	213	174	86	54	43	70	97	181	218	179	92	56	41
21	53	119	159	121	55	22	5	25	59	123	157	128	57	23	6

Abbildung 13: Kleine »Superzelle« mit vier Rasterpunkten

PostScript berechnet die Werte auch nur einmal und speichert nur noch die besagten Zahlen in der berüchtigten – berüchtigt deshalb, weil ich den Begriff früher schon öfter gehört habe, ohne mir etwas darunter vorstellen zu können – *threshold matrix* ab oder besser *threshold array*, wie Adobe sich ausdrückt. Die Grenzen dieses threshold arrays machen die wesentlichen Schwierigkeiten bei etwas ausgefallenen Rastern aus. So ist die Größe dieses threshold arrays leider begrenzt. Außerdem wird die Spotfunktion nur ein einziges Mal abgearbeitet, was bedeutet, daß irgendwelche zufallsmäßig erzeugten Raster von vorneherein ausscheiden. An dieser Stelle möchte ich schon einmal auf meinen nächsten Artikel hinweisen, wo ich zeigen werde, wie man den Rastermechanismus austricksen und trotzdem Zufallsraster erzeugen kann, allerdings mit dem Wermutstropfen einer ziemlich langsamen Verarbeitung.

Alles ganz einfach – und doch wieder nicht

Ist ja alles ganz einfach, wenn man alles mit ein paar Befehlen bestimmen kann. So hatte sich das der Erfinder von PostScript auch gedacht. Aber leider hatte er erst mal an Schwarzweißbilder gedacht und sich nicht überlegt, daß bei Überlagerung von mehreren Farben *Moirés* entstehen. Damit bei einem normalen Raster keine Moirés entstehen, müssen die Winkel penibel eingehalten werden und auch die Rasterweite muß genau stimmen. Und hier liegt das große Problem. Im mathematischen Modell des PostScript-Textes stimmt auch alles ganz genau. Aber da stößt der Interpreter an seine Grenzen oder besser gesagt an die Grenzen des Ausgabegerätes. Mit einem unendlich feinen Ausgabegerät könnte jedes Raster wirklich perfekt dargestellt werden. Aber unendlich genau klingt auch nach unendlich teuer und scheidet somit als Lösung aus. Leider habe ich zur Darstellung der Rasterungsproblematik kein so schönes Programm zur Verfügung wie für die Spotfunktion, aber indem ich das Programm etwas mißbrauche, kann ich schon einiges zeigen. In Abbildung 13 sieht man vier Rasterpunkte in einer Rasterzelle, was die einfachste Möglichkeit dieser Art von Raster ist. Die Spotfunktion ist etwas zu lang, um sie in der Graphik unterzubringen. Ich möchte sie den Interessierten aber nicht vorenthalten, sie lautet:

```
/spotfunc {abs 2 mul 1 sub abs/Y1
exch def abs 2 mul 1 sub abs/X1
exch def X1 Y1 add 1 ge
{1 X1 sub dup mul 1 Y1 sub dup mul
add −1 add}
{X1 dup mul Y1 dup mul add 1 exch
sub}
ifelse} bind def
```

In Abbildung 14 sieht man noch einmal dieselben Rasterpunkte, aber in einem Winkel von 10 Grad verwinkelt. Wenn man sich nun diese Zellen nebeneinander vorstellt, sieht man leicht, daß nicht einfach gewinkelt ist, sondern daß sich Sprungstellen ergeben. Nun kann man sich leicht vorstellen, daß man eine sehr große Rasterzelle bräuchte, um einen einigermaßen nahtlosen Anschluß zu bekommen. Die Spotfunktion ist ähnlich der, die mit der Abbildung 13 erzeugt wurde. Allerdings wird gewissermaßen noch eine Rotation vorgeschaltet. Hier auch die Funktion mit eingebautem Winkel:

```
/winkel 10 def
/spotfunc{/y1 exch def /x1 exch def
/x2 x 1 winkel cos y1 mul winkel sin
mul sub def /y2 x 1 winkel sin mul
y1 winkel cos mul add def x2 y2
abs 2 mul 1 sub abs /Y1 exch def abs
2 mul 1 sub abs /X1 exch def X1 Y1
add 1 ge
{1 X1 sub dup mul 1 Y1 sub dup mul
add −1 add}
{X1 dup mul Y1 dup mul add 1 exch
sub}
ifelse} bind def
```

Am Anfang wird lediglich der Winkel definiert. Das hat einen Vorteil, wenn man nur den Winkel ändern will. Beigefügt habe ich noch die Abbildung 15, in der ein Winkel von 30 Grad verwendet wurde. Hier passen die Zellen anscheinend zusammen. Ich bitte dabei, darüber hinwegzusehen, daß eigentlich je vier aneinandergrenzende Superzellen an den Ecken noch einen Punkt bilden müßten, was zu aufwendig geworden wäre. Aber wenn man genau hinschaut, wird man außerdem feststellen, daß die Mittelpunkte der Rasterpunkte nicht rechtwinklig zueinander stehen. Das normale PostScript arbeitet zwar nicht genauso, aber das Problem bleibt dasselbe. Große Zel-

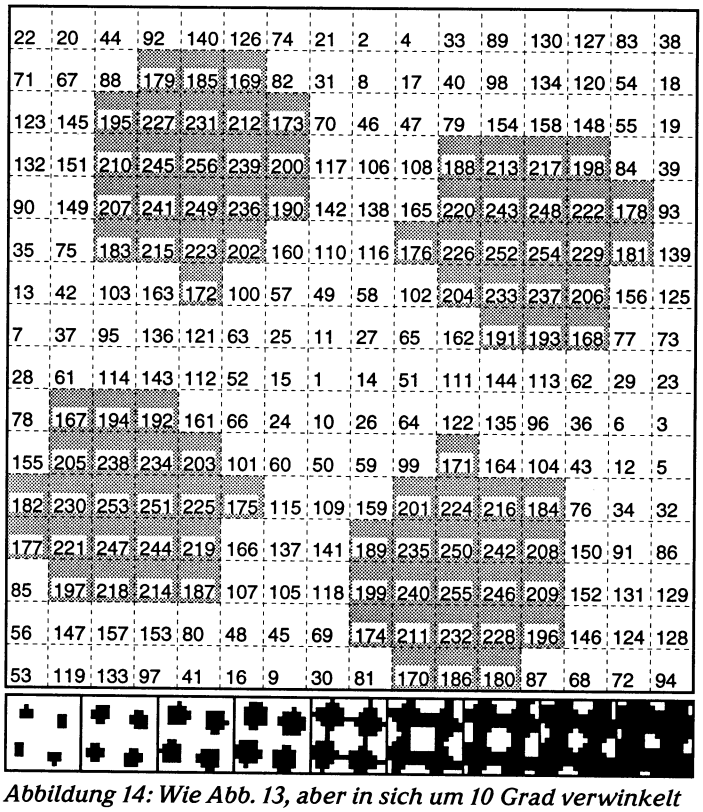

Abbildung 14: Wie Abb. 13, aber in sich um 10 Grad verwinkelt

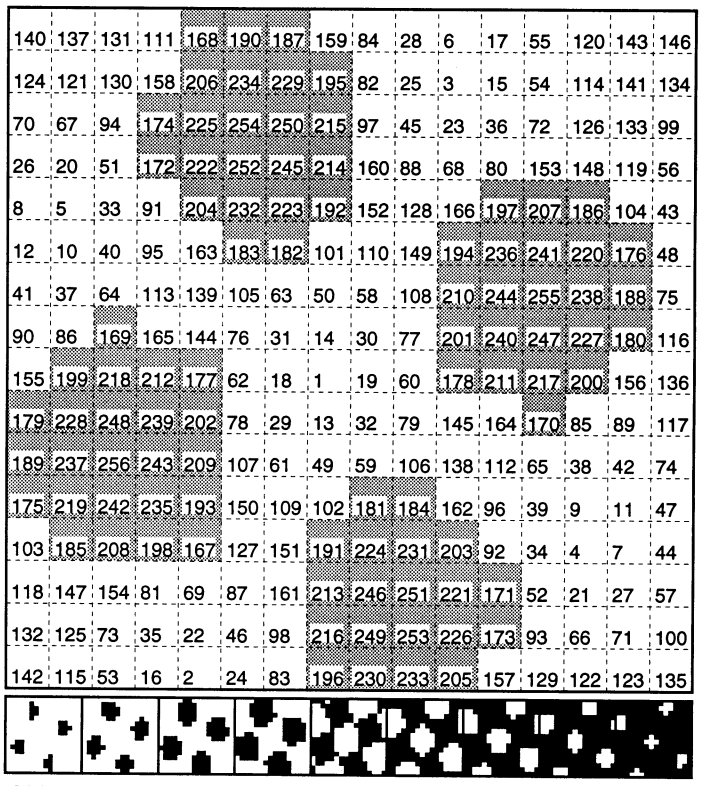

Abbildung 15: Wie Abb. 14, aber in sich um 30 Grad verwinkelt

len mit mehreren Rasterpunkten, sogenannte Superzellen, werden aber bei den neueren, verbesserten Rastern benutzt. Je größer aber die Rasterzelle, desto mehr Speicherbedarf und desto länger die Rechenzeit. Einen 15-Grad-Winkel darzustellen, bleibt aber trotzdem immer nur eine Annäherung. Mit genügend großen Zellen kommt man aber schon ziemlich nahe heran. Um den restlichen Fehler noch auszugleichen, müssen bestimmte Rasterkombinationen gewählt werden, so daß sich die Fehler möglichst nicht gegenseitig addieren, sondern vermindern. So arbeiten alle Raster mit den klingenden Namen: Accurate-Screening, Balanced Screening und HQS. Um es mal überspitzt auszudrücken, handelt es sich um nichts anderes als ein riesen »Gefummle«, um die unvermeidlichen Fehler auszugleichen. Und das Ganze ist dann auch noch nach eigenen Aussagen empirisch ermittelt, was nur etwas vornehmer dafür steht, daß man die Effekte auch nicht vorhersagen konnte, sondern ausprobieren mußte, welche Rasterkombinationen nun die besten sind. Mit Hilfe der Superzellen könnte man auch noch das berüchtigte Dithering erzeugen. Der Begriff heißt eigentlich Zittern. Es handelt sich um das einfache Problem, daß, wenn zwei Flächen aneinanderstoßen, die in sich genau denselben Tonwert haben und sich voneinander nur ganz schwach unterscheiden, dieser Unterschied manchmal durch bestimmte Effekte deutlicher wahrgenommen wird als er von der Vorlage erscheint. Berühmtestes Beispiel für diesen Effekt ist ein Graukeil. Auch wenn sich die einzelnen Stufen nur um weniger als 1% Flächendeckung unterscheiden, nimmt man die einzelnen Stufen oft noch deutlich wahr. Als Gegenmaßnahme wirft man in den Grenzbereichen die Grauwerte etwas durcheinander.

Aber trotzdem ...

Innerhalb der Grenzen, die von PostScript vorgegeben sind, läßt sich eine ganze Menge machen. Wenn man allerdings so nahe wie möglich an der herkömmlichen Rasterung bleiben will, dann sind die neuen Raster der einzig richtige Schritt. Aber natürlich würde ich keine drei Artikel schreiben, wenn ich nicht der Meinung wäre, daß es doch einen Sinn hat. Im nächsten und letzten Teil dieser kleinen Serie will ich eine konkrete Folgerung aus allem bisher auch im ersten Teil Gesagten ziehen und ein etwas anderes Raster vorstellen. Außerdem habe ich ja schon einmal anklingen lassen, daß man den Rastermechanismus von PostScript auch umgehen kann. Man hat zwar gigantische Rechenzeiten, aber es geht. Und so ein Zufallsraster ist ja eine interessante Sache, zum Ausprobieren allemal. Und vielleicht ist es ja eine Anregung, mal einen PostScript-Interpreter umzuprogrammieren. Es gibt nämlich einen Shareware-PostScript-Interpreter für UNIX-Systeme von einem Projekt namens GNU. Leider hatte ich bisher selbst noch keine Zeit dazu. Oder es bliebe zu wünschen, daß PostScript eine Schnittstelle zur Maschinensprache des Interpreters zur Verfügung stellt. Das mag illusorisch klingen, aber im Display-PostScript gibt es eine Schnittstelle zur Programmiersprache C über den Präprozessor PSWrap. Allerdings dient dieser nur dazu, PostScriptCode von C aus aufzurufen. Insgesamt wäre es aber der einzig vernünftige Weg, den Interpreter oder den RIP umzuprogrammieren, anstatt wie hier demonstriert im Programmtext herumzubasteln, wenn es sich auch nur um das Ändern von ein bißchen Programmtext handelt. Wenn man nur im Programmtext ändern kann, ist aber auf Dauer der PostScript-Code, der ja eigentlich das Verständigungsmittel ist, in seiner Einheitlichkeit in Gefahr.

Vielleicht ist hier noch ein kleiner Ausblick auf PostScript Level 2 sinnvoll. Das Accurate-Screening ist im neuen PostScript ganz normal mit enthalten, wie es auch von den anderen Anbietern normalerweise ohne Aufpreis mitgeliefert wird. Im Level 2 kann auch das threshold array direkt programmiert werden, was eventuell bessere Rasterprogrammierung zuläßt. Außerdem läßt sich wahrscheinlich die Größe des threshold arrays verändern, wodurch man selbst Superzellen programmieren kann. Das Problem dabei ist, daß es bis jetzt erst sehr wenige Geräte mit Level 2 gibt und man deshalb auf die Handbücher von Adobe angewiesen ist. Dort sind dann nur die einzelnen Befehle erklärt. Daraus kann man aber nicht immer schließen, wie die Befehle kombiniert wirken, insbesondere, wenn man, wie man es bei Programmierung von Rastern tun muß, ziemlich in den Extremen der Sprache arbeitet.

(Beitrag wird fortgesetzt)

SCHLÜSSELWÖRTER:
Bildverarbeitung
Moiré
PostScript
Rasterpunktform
Rastertechnik
Spotfunktion
Superzellen