C. BITMAPPED GRAPHICS
(Draft, Sommer 1999)

Vorbemerkung: Der folgende Versuch ist aus einem Seminar „Punkte, Zeilen, Spalten. Von der Weberei zur Computergraphik“ entstanden, das im Wintersemester 1998/1999 von Birgit Schneider (Forschungsstelle Technisches Bild, Helmholtz-Zentrum für Kulturtechniken der HU Berlin) und Peter Berz (Seminar für Ästhetik der HU Berlin) gehalten wurde. Es ist der dritte von drei Artikeln, von denen bislang die ersten beiden erschienen sind: Birgit Schneider, Peter Berz, Bildtexturen. Punkte, Zeilen, Spalten. I. Textile Processing (B. Schneider). II. Bildtelegraphie (P. Berz), auf: http://waste.informatik.hu-berlin.de/mtg/mtg4/Schneider_Berz/textil.html ; gedruckt in: Intervalle 5. Mimetische Differenzen. Der Spielraum der Medien zwischen Abbildung und Nachbildung (hg. Sabine Flach, Georg Christoph Tholen), Kassel 2002, S. 181 - 219.

Inhalt

0.

"In bitmapped graphics, each dot on the screen (called a *pixel*, or picture element) corresponds to one or more bits in memory. Programs modify the display simply by writing to the display memory. Bitmapped graphics are also referred to as raster graphics, since most bitmapped displays use television-type scan line technology: the entire screen is continually refreshed by an electron beam scanning across the face of the display tube one scan line, or raster, at a time. The term bitmapped graphics (or memory-mapped graphics) is more general, since it also applies to other dot-oriented displays, such as LCD screens. We assume you are familiar with the basic principles of bitmapped graphics."[1]

[1] Adrian Nye, Xlib Programming Manual (The Definitve Guides to the X Window System, vol. 1), O'Reilly 1988/1995:

1. Historische Einleitung: Viewing tubes

Was im Handbuch für *Xlib*, der Bibliothek des mächtigen Graphiksystems *XWindow* [2], nur im Keller der Anmerkungen haust, eröffnet medientheoretisch Anschlüsse an eine Geschichte der Zeilen- und Spaltenbilder seit der Weberei. Schon die Trennung von *screen* und *memory*, Bildschirm und Code ist ein Standard, der nicht immer so unumschränkt herrschte wie *Xlibs* Axiomatik es glauben läßt. Die Nachkriegsgeschichte des Computers, die Speichertechniken sucht, aber noch keine graphischen Ausgabegeräte, vulgo Bildschirme, kennt, hebt - folgt man einer Anregung Andre Reifenraths[3] - diese Trennung für technisch folgenlose, aber theoretisch interessante fünf Jahre auf.

Nach ENIAC, EDVAC, Turings Colossus und Zuses Z's sucht die Technik des *stored programming* schnelle, raumsparende und präzise Speicher. Das führt umstandslos zurück in Weltkriegstechnologie. 1946 war nicht nur "über Nacht eine Masse von Radarexperten mit endlosen Problemen, für die sie Lösungen suchten, in eine Expertenmasse mit endlosen Lösungen und keinen Problemen verwandelt".[4] Zum Sperrmüll aus zehn Jahren Luftkrieg gehören auch Tausende von Radarröhren. Seit Ende 1945 arbeiten auf ein Begehren John von Neumanns zwei Ingenieure der *Radio Corporation of America, RCA,* an Radarröhrenrecycling.[5] Ihre Idee, eine Radarröhre zum elektrostatischen Speicher umzubauen, "a tube for selective electrostatic storage", *selectron*, antwortet auf ein Problem, an dem bereits der ENIAC laborierte.

Die flip-flops der *vacuum tube circuits*, die alle arithmetischen Operationen des Computers ausführen, also der Prozessor, laufen um ein Vielfaches schneller als jeder verfügbare Speicherzugriff. Die Zeit für Lesen und Schreiben von Magnetbändern (17 Millisekunden), *mercury delay lines* (0.5 - 1 Millisekunde) oder gar Lochkarten wird von elektronischen Röhren-Geschwindigkeiten des Prozessors um ein Vielfaches unterboten.

S. 4, Anmerkung.

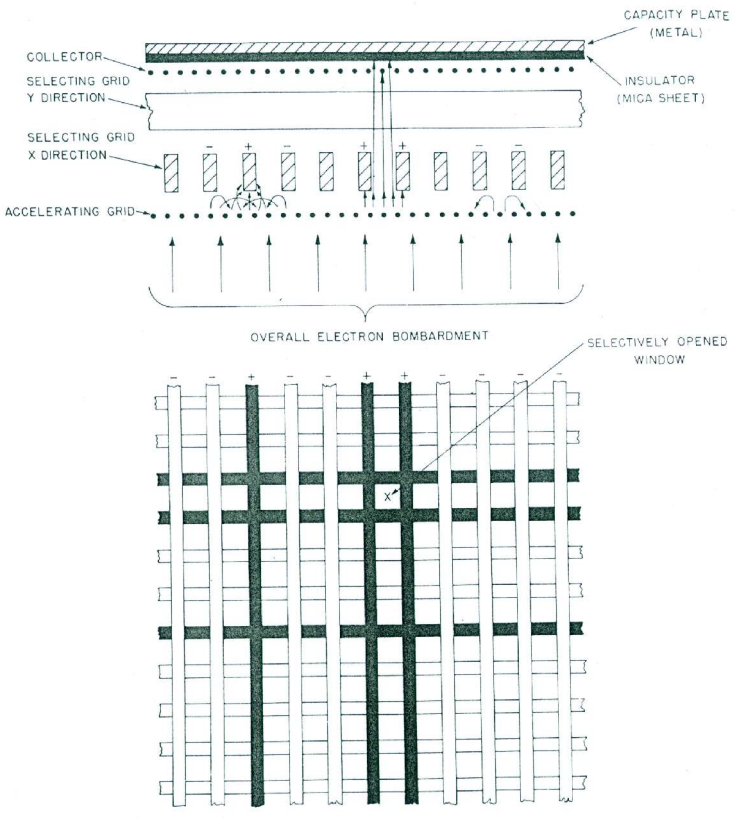
[2] Dem Windowstandard unter UNIX alias LINUX. Die Mächtigkeit oder Macht von XWindow beruht erstens auf seiner Unabhängigkeit von bestimmten Plattformen, also Betriebssystemen; zweitens seiner Unabhängigkeit von spezieller Software für spezielle Benutzeroberflächen, also bestimmte "Windowmanager"; drittens darauf, daß es schon intern als Netzstruktur läuft und darum extern in Netze paßt: die *client/server*-Struktur ermöglicht, daß Anwendungen Rechenkapazitäten irgendwo im digitalen Weltnetz nutzen, sie ermöglicht *distributed processing* von Graphik.

[3] Vgl. Reifenrath, 1998. Zur Diskussion von Reifenraths Darstellung und Folgerungen vgl. weiter unten.

[4] Williams, ?: 1.

[5] Pugh, 1984: 15, Goldstine, 1972/1993: 309, über den Kooperationsvertrag von v.Neumans IAS und RCA, vgl. ebd: 242.

Die naheliegende Idee, flip-flop-Schaltungen mit zwei Vakuumröhren als Speichertechnik einzusetzen, wurde im ENIAC zwar für "very small memories called registers" eingesetzt.[6] Für größere Speicher aber, wie die von v. Neumann anvisierten 160 000 bits (4000 words à 40 bits oder 20 KB), ist ein Speicher, der für jedes bit zwei Vakuumröhren braucht, eine hoffnungslose, modulare Monstrosität. Dagegen würden 256, 1024 oder mehr bits auf einer Röhre die Zugriffszeit in den Mikrosekunden-Bereich senken. Techniken, dieses Versprechen durch Bildröhren aus der Radartechnik einzulösen, stehen, heißt das, Ende der 40er Jahre hoch im Kurs.



PRINCIPLE OF SELECTRON

Figure 1

Die komplizierten *selectron*-Röhren des RCA bringen vor die elektrostatisch aufladbare Fläche des Bildschirms "zwei Netze (oder Gitter, engl. *grids*) aus Stäben in gleichem

[6] Pugh, 1984: 35, auch im IBM 604 werden sie verwendet.

Abstand, die im rechten Winkel zueinander stehen".[7] Die Rasterzwischenräume, die Goldstine/Rajchman *windows* nennen und Pugh *eyelets*, also Gucklöcher, können durch Zeilen/Spalten-Adressierung der Stäbe positiv oder negativ geladen werden. Sie sind dann für einen auf das Gitter treffenden Elektronenstrahl entweder geöffnet oder geschlossen. Die statistischen Ereignisse am Gitter jeder Elektronen-Röhre sind damit so selektiv gewendet als wäre Maxwells Dämon persönlich am Werk. Dringt der Strahl durch ein geöffnetes Fenster, geht er ungehindert bis zur Phosphorschicht des Bildschirms durch und lädt dort eine Stelle elektrostatisch auf. Durch getrennte Ansteuerung jedes Augenlochs oder Fensters im Gewebe wird der Bildschirm als Matrix adressierbar. Jeder einzelne Punkt kann als Speicherplatz mit elektronischer Geschwindigkeit beschrieben oder ausgelesen werden[8] und die traurige Gleichung der ENIAC-Zeit mit ihren 18 000 tubes: 1 Röhre = 1 Speicherplatz = 1 bit ist trotzdem obsolet.

Die "Schönheit" (Goldstine) eines britischen Konkurrenzsystems zu *selectron*, das auf den Namen des Radarexperten Frederic Calland Williams vom *Telecommunications Research Establishment TRE* hört, besteht vor allem darin, Weltkriegs-Radarröhren unverändert so zu verwenden, wie sie aus dem Kriegseinsatz kommen. Außerdem ist der Speicherplatz der sogenannten *Williams Tubes* nicht durch die Hardware eines Metallgeflechts auf magere 256 binäre Stellen begrenzt. Sie speicheren 1024 binary digits und mehr.

Als Radar-Ingenieur hatte F.C. Williams während des Kriegs das Verfahren der "Festzielunterdrückung" mit entwickelt. Da die steigende Effektivität der Radarabtastung in den letzten Kriegsjahren nicht nur mehr Daten, sondern auch mehr Schrott auf die Radarschirme zaubert, wird die Trennung von Signal und Rauschen immer essentieller. Würde etwa, so Williams' schlichte, elektrotechnische Lösung[9], jeder Umlauf, also jeder *scan* des Radarstrahls auf dem runden Bildschirm, gespeichert und dann vom nächsten *scan* subtrahiert, so blieben nur die Abweichungen der beiden Bilder übrig. In genauer Umkehrung früher Photographien mit ihren endlosen Belichtungszeiten wären dann nur noch die bewegten Elemente auf dem Radarschirm zu sehen. Die befremdliche Idee, eine analoge Übertragungstechnik namens Radar zum Speichern digitaler Muster einzusetzen, entsteht, heißt das, im Kontext von *moving target identification*.

[7] Goldstine, 1972/1993: 309.

[8] SELECTRON wird vor allem bei IBM verwendet, mit den taktischen Vorteil: der Output ist um vieles stärker als bei der Williams Röhre, also störungsresistenter (vgl. Gruenberger, 1979: 62).

[9] Amerika geht das Problem theoretischer an: Claude Elwood Shannons Nachrichtentheorie behandelt das Problem mathematisch so allgemein, daß daraus eine ganze *Communication Theory* werden kann.

1946 wird Williams zusammen mit dem Radarexperten Tom Kilburn an die Universität Manchester berufen zur Gruppe Maxwell Hermann Alexander Newmans, die am Bau eines Computers arbeitet. Im September 1948 stößt auf Veranlassung Newmans die größte anzunehmende britische Autorität in Sachen Computer zur Manchester Gruppe: Alan Mattison Turing himself.[10]

> "Tom Kilburn and I knew nothing about computers, but a lot about circuits. Professor Newman and Mr. A.M. Turing in the Mathematics Departement knew a lot about computers and substantially nothing about electronics. They took us by the hand and explained how numbers could live in houses with addresses and how if they did they could be kept track during a calculation."[11]

Turings Einführungskurse in die Theorie der Adressierung aber wollen Elektronik werden und da entsteht ein Problem. Um aus dem Bildmedium Elektronenröhre ein digitales Speichermedium zu machen, ist eine Systemdifferenz zu überwinden. Radarschirme, Iconoscops, TV sind "Integrationen über die Zeit".[12] Praktisch keine der bisherigen Anwendungen von Bildröhren hatte, so Williams, strikte Fehlerfreiheit, "absolute freedom of malfunction" zur Bedingung. "In communication a missed word could be repeated. In radar if you missed one echo you would probably get the next."[13] Was am TV ein einigermaßen scharfes und ruhiges Bild scheint, kann in Wirklichkeit voller "Ausfälle und fehlender bits" sein, das heißt: Rauschen.[14] Kinogeschichte, die mit der stroboskopischen Integration über die Zeit von Bild zu Bild begann, greift im elektronischen Bild auf die Ebene der Bildpunkte durch. Doch: "When you get down to saying 'what's on that little bit there?'", dann ist Integration über die Zeit und "a braod brush thing like TV" die systematisch falsche Basistechnologie. Denn sie erlaubt nicht, was nach Turings Lektionen in Manchester der Kern der Sache ist: die Adressierung eines einzelnen Speicherplatzes.

Williams und seine Assistenten machen als Techniker aus dieser Systemdifferenz erst einmal ein quantitatives Problem. Um eine Bildröhre zum bitgenauen Speicher umzubauen, muß die Qualität der Röhren gesteigert werden. Angelieferte Röhren werden also strengen Tests unterworfen, bis noch ganze 19% als brauchbar übrig bleiben.[15]

---

[10] Zur Geschichte des Manchester Computers, vgl.: Goldstine 1972/1993: 247 f., Campbell-Kelly, 1980: 131 - 134, Williams, 1975.
[11] Williams, **?**: 2.
[12] Sale, 1998.
[13] Williams, ?: 1.
[14] Sale, 1998.
[15] Goldstine, 1972/1993: 310, für die Praxis des IAS.

Sodann trägt man Sorge, daß die elektrostatisch aufzuladende Phophorschicht am Bildschirmglas so rein wie möglich ist[16], und außerdem wird die Röhre durch Eisengehäuse gegen Erschütterung oder elektromagnetische Felder der Umgebung isoliert.[17] Schließlich aber reduziert man die Ausfälle durch drastische Reduzierung der Auflösung des Bildschirms: die Williams Tubes in v. Neumann/Goldstines IAS-Computer etwa nutzen pro Röhre ganze 32 mal 32 Punkte mit nur zwei Zuständen - elektrostatisch geladen/nicht geladen; ein Fernsehbild auf der gleichen Röhre stellt für jeden von 500 mal 500 Punkten kontinuierliche Schwarz-Weiß-Übergänge dar.

Williams Speicherröhren sind also digitale Zweckentfremdung einer analogen Bildübertragungs-Technik. Sie arbeiten an *A/D-conversion*, an der Wandlung analoger in digitale Signale, nicht weniger als die Bildtelegraphie des 19. Jahrhunderts an der digitalen Wandlung des analogen Mediums Photographie. Williams/Kilburns Paper von 1948 liefert denn Beschreibungen digitaler Dispositive, deren Elementarität, auch wenn Turing erst im September 1948 nach Manchester kommt, seine Handschrift tragen dürften. Zwischen einer Eröffnung über *computing machines*, die nicht denken[18], einer Einführung in binäre Zahlensysteme und der Philosophie digitalen Lesens, Schreibens und Löschens[19] tauchen, wie verschoben auch immer, die Grundzüge eines elektronischen Rasters aus adressierbaren Punkten, Zeilen und Spalten auf. Ihm wäre die Frage zu stellen, ob es ein Bild ist oder nicht.[20]

1.1 Der Punkt

Physikalisch gründet Williams' Speichertechnik diesseits aller quantenphysikalischen Leuchteffekte elektronischer Bildschirme[21] auf zwei Vorgängen aus der Welt der Elektronen. Treffen die Elektronen des Röhrenstrahls auf die Phosphorschicht, die sich seit Ferdinand von Braun vor dem Glas einer Bildschirm-Röhre befindet, bilden sie dort

[16] IBM kämpft für seine *Serie 700*, die mit Williams Röhren läuft, gegen Schmutz in der Luft und, nach Umzug in die freie englische Natur, gegen einzelne Blütenpollen, die mit einzelnen bits konkurrieren (Kilburn, XX: 2 f.).
[17] Vgl. Goldstine, 1972/1993: 311.
[18] "It should be stated at once that a computing machine cannot 'think'." (Williams/Kilburn, 1949: 184) Denn jedes Problem müsse, bevor es errechnet werden kann, "externally (i.e. outside the machine)" in einfachste arithmetische und logische Operationen zerlegt werden. Daraus ist dann eine Tabelle von Instruktionen zu konstruieren, in der jede Instruktion ebenso wie jedes Datum eine binär gespeicherte Adresse hat. Williams/Kilburns Ausgangsbehauptung ist so wenig Turing wie ihre Begründung Turing ist.
[19] Etwa: "'Erasing', of course, implies that information is erased from a store, but in its preferabele form it is really a superseding process in that a word may be written into an occupied address, deleting the word already there." (Williams/Kilburn, 1949: 184).
[20] Eine Frage, die Andrej Reifenrath leider übergeht.
[21] Vgl. weiter unten.

keinen Elektronenhaufen, sondern eine Art Krater, "termed a 'well'".[22] Dieser Krater, *well* oder Punkt oder *spot*, "the bombarded area of the screen", ist nach Auftreffen des Strahls leer von Elektronen und hat darum positive Spannung. Dazu kommt ein zweiter Vorgang: Ein Elektronen-Bombardement in bestimmter Nähe des entleerten Kraters, füllt diesen wieder mit Elektronen auf, die positive Spannung verschwindet. Da bei Williams kein Auge ein Bild sehen, sondern eine Maschine einen Speicher beschreiben und lesen soll, liegt vor dem Glas des Bildschirms ein zweiter Schirm, *a conducting screen*, der eben diese positiven oder negativen Spannungen der *wells* aufnimmt und über einen Verstärker zu positiven oder negativen Signalen, 1 oder 0 macht.

Auf *empty wells* und *filled wells* und ihre Auswertung als Signale - nicht als Bildpunkte - baut Williams die Logik eines binären Speichers. Von Loch/Nicht-Loch auf Jacquards Pappkarten unterscheidet sie sich schon darum, weil jedes *storage element* allein als doppelter Punkt, als zwei benachbarte *wells* funktioniert. Alles Schreiben und Lesen in Williams' binär codiertem Speicher beginnt mit dieser Minimaldifferenz, die am Ende nicht mehr Derrida wäre. Denn ihr Grund ist das Elektronische selbst von elektronischem Lesen und Schreiben. Der Elektronenstrahl tastet nicht nur ab wie die Nadeln bei Jacquard, sondern er verändert den Zustand des Speichers. Ein und derselbe Strahl liest und schreibt zugleich. Der vom Gitter der Röhre an- und abgeschaltete Elektronenstrahl bombardiert den ersten Speicher-Punkt (leert ihn oder findet ihn schon leer vor) und kann dann den Nachbarpunkt bombardieren (um den ersten zu füllen) oder nicht bombardieren (um den ersten leer zu lassen). Die Folge *beam ON - beam OFF* schreibt also eine 1, wenn eine Eins durch ein *empty well* und positive Ladung dargestellt ist; die Folge *beam ON - beam ON* schreibt eine 0, wenn eine Null durch ein *filled well* und negative Ladung dargestellt ist. Diesseits aller komplexen, physikalisch-elektronischen Bedingungen machen dann genau vier Fälle die Schreib/Lese-Logik von Williams' Speicher-Röhren aus:

[22] Williams/Kilburn, 1949: 186. Der Grund: die sogenannte *secondary emmission*.

				Output	
	1	0	→	0 1 :	–
beam	on	on			
	0	0(1)	→	0 1 :	–
beam	on	on			
	0	0(1)	→	1 0 :	+
beam	on	off			
	1	0(1)	→	1 0 :	+
beam	on	off			

(Ein *empty well* wäre als 1, ein *filled well* als 0 dargestellt; links vom Pfeil steht der Ausgangszustand, rechts das Ergebnis; eine 1 in Klammern heißt: das Ergebnis verändert sich nicht, egal ob der zweite spot eine 1 oder eine 0 ist.)[23]

Zwei Tatsachen aber stellen Schreiben und Lesen binärer Punkte auf elektrostatisch aufladbaren Flächen ebensosehr in Frage wie sie den Unterschied zu allem anderen Lesen und Schreiben sichtbar machen. Würde der Lesestrahl der einfachen Folge *beam ON - beam ON* gehorchen, dann wäre das Ergebnis des Lesens in allen Doppelspots gleich: erster *well* voll, zweiter leer. Der Speicher wäre beim Lesen gelöscht worden. Die zweite Tatsache führen Williams' Kritiker als Unmöglichkeit elektrostatischer Speicher überhaupt ins Feld: die Verteilung der Spannungen auf der Fläche wird nur einige Zehntel Sekunden aufrechterhalten, bevor sie durch *leakage* der Oberfläche verschwindet. Anders als Lochkarten und flip-flops ist ein elektrostatischer Speicher ein flüchtiges Kurzzeitgedächtnis: er muß in einem bestimmten Takt regeneriert werden.

Doch Williams' Doppelspots finden in der Zeit statt - spot 1 zur Zeit t_1, spot 2 zur Zeit t_2 - und darum kann auch das Prinzip der Regeneration einfach sein. Das Signal zum Bombardement, das an das Steuergitter der Elektronenröhre getriggert wird, wird mit dem Signal rückgekoppelt, das der Verstärker beim Lesen empfängt. Ist das Signal negativ, wird für den Zeitpunkt t_2 der Strahl ausgeschaltet und spot zwei wird nicht bombardiert; ist es positiv, wird er eingeschaltet und spot zwei wird bombardiert. "... the system will be regenerative in that it will immediately re-write everything it reads."[24] Die Konsequenzen solcher Anordnungen für eine Theorie der digitalen Schrift wären erst noch zu ermessen.

[23] Vgl. auch die axiomatische Formulierung bei Williams/Kilburn, 1949: 188 a und b.
[24] Ebd.: 189.

1.2 Die Matrix

Die Logik von Lesen und Schreiben eines einzelnen Speicher-Punkts auf dem Bildschirm ermöglicht noch nicht dessen Adressierung. Williams Phosphor-*wells* wären als Trichterfeld so wenig adressierbar wie die "chaotischen Zonen" (Jünger) der WW-I-Schlachtfelder, für die Ludendorff noch im September 1918 "unauffällige, schachbrettförmig liegende Anklammerungspunkte" aus dem Hut zaubern will.[25] Punkte oder Doppelpunkte in Williams' Röhre liegen darum von Anfang an in den Zeilen und Spalten einer 2-dimensionalen Matrix. Zwar würde der kreisförmige Bildschirm einer Radar-Röhre ganz andere optimale Ausnutzungen verlangen. Aber verschiedene, wenn auch inkrementell wachsende und schrumpfende Zeilen- und Spaltenlängen würden vor neue Probleme stellen. "Accordingly a rectangular array has been chosen."[26] Das horizontale Ablenksignal überlagert sich also mit dem Signal zum An- und Abschalten des Elektronenstrahls und einem *Y-shift generator*, der am Ende jedes horizontalen Durchlaufs einer Zeile einen shift senkrecht und in Spaltenrichtung zur Länge der Zeile ausführt.[27]

Allein ihre matrizenförmige Anordnung macht aus Speicherelementen adressierbare Speicherelemente. Die allgemeinste Beschreibung von Williams' System kann darum schlicht ausfallen:

> "The digits are represented by charge distributions which exist on small areas of a c.r.t. screen, the carge distributions being arranged in the form of a two-dimensional array. This array is produced by a television type of raster, in which the digits of a line, and the lines of the raster, are scanned sequentially, each digit corresponding with a 'picture element'."[28]

[25] Die Abwehr im Stellungskriege. Vom 20. September 1918. In: Urkunden der Obersten Heeresleitung über ihre Tätigkeit 1916/1918 (hg. von Erich Ludendorff), XXIV. Militärische Schriften. 4. Aufl. Berlin 1922: 613, Ziff. 31.
[26] Williams/Kilburn, 1949: 188.
[27] Ebd..
[28] Williams/Kilburn, 1949: 184. Über den Gebrauch des Wortes *picture element* in der Fernseh- und Radartechnik vor und um 1948 sind vorerst nur Spekulationen möglich (vgl. auch nächste Anmerkung). Daß Williams/Kilburn es in Anführungszeichen setzen, mag der Tatsache geschuldet sein, daß für die Autoren das Raster der Speicherröhre nur in ungewöhnlicher Weise als Bild ansprechbar ist.

1.3 Das Bild

Am 21. Juni 1948 arbeitet die *baby machine* des Manchester Mark I zum ersten Mal erfolgreich mit einem Williams Speicher 32 Zeilen à 32 *spots*; die verbesserte Version vom Frühjahr 1949 mit vier Williams Röhren, von denen jede eine Matrix von 32 Zeilen à 40 binäre Speicherelemente alias bits speichert, 40 bits = 1 word, macht 128 words oder 5120 bits.[29] Die in *words* oder *lines* - eine Zeile gleich ein Wort[30] - formatierten Speicherelemente aber kommen in Manchester schließlich zu ihrem Usprung als *picture elements* zurück.

Denn schon der erste funktionierende Williams-Speicher nutzt Röhrentechnologie auch rückwärts gewandt. Auf die Speicherröhre wird eine *monitor*, *display* oder *viewing tube* gesetzt, die jede Speicherstelle auf einem Bildschirm sichtbar macht. Die Display-Röhren aber sind mit den Speicherröhren technisch identisch.[31] Denn jedes der wiederholten Bombardements der Phophorfläche lädt nicht nur ein Speicherelement elekrotstatisch auf und erzeugt ein Signal. Jeder Elektronenbeschuß ergibt auch quantenphysikalische Lumineszenz-Effekte, die eine positiv geladene Speicherstelle zum hellen Punkt, eine negativ geladende zum dunklen Punkt machen.[32] Die Röhre gibt dann ein Bild aus Punkten, Zeilen, Spalten zu sehen, ein Raster aus 32x40 hellen und weniger hellen Punkten. Sie stellen nichts dar als den Zustand des Speichers selbst: helle Punkte Eins, dunkle Punkte Null.[33]

---

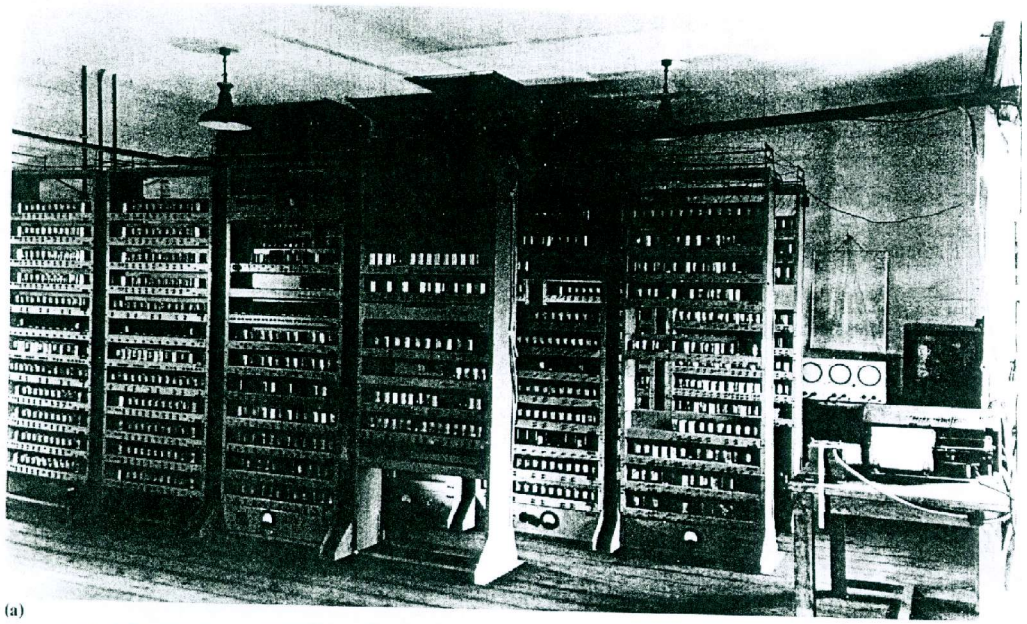
[29] Dazu kommt eine *magnetic drum*, ein Magentband-Speicher. Manchester, England, arbeitet seriell, Princeton, Amerika, (also v. Neuman und IAS) arbeitet nach der alten Maßgabe des ENIAC parallel: ein word = 40 bit mit 40 Williams Röhren à 32x32 bit, also insgesamt 1024 words und für jedes word wird ein bestimmtes bit in jeder Speicherröhre adressiert.

[30] "Incidentally, *line* is almost always used in preference to *word* in the Mark I literature; a long line was 40 bits, a short line 20 bits, and a line usually meant a short line." (Campbell-Kelly, 1980: 136).

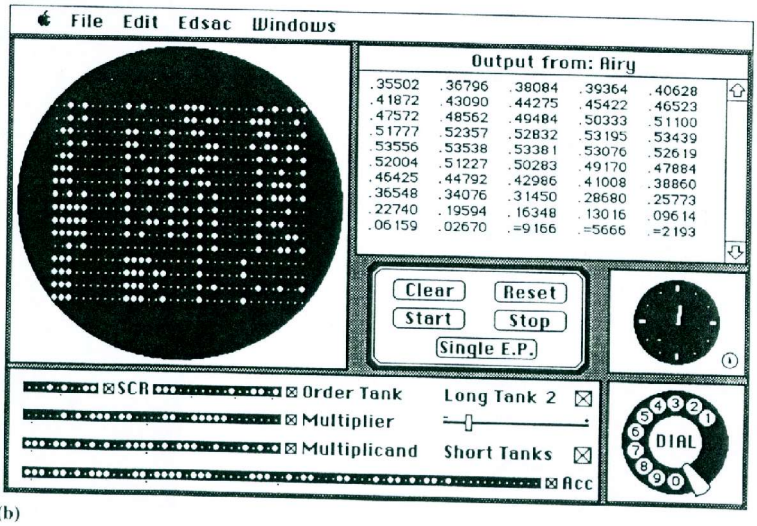
[31] Die IAS-Maschine kann über einen 40-Stufen-Schalter jeweils eine der 40 Röhren darstellen (vgl. Goldstine, 1972/1993: 312).

[32] Vgl. auch weiter unten.

[33] Zunächst setzt man auf telegraphische Darstellung mit Punkten und Strichen, bald auf ein*focus/defocus*-System (Sale, 1998).



(a)



(b)

Figure 1. The EDSAC, then and now. (a) A photograph of the EDSAC taken shortly after its completion in May 1949. The left three quarters of the picture show the main racks of the arithmetic unit, control, and memory. The input/output equipment (a paper-tape reader and teleprinter) can be seen on the table at the right. Three of the monitor tubes can be seen to the rear and right of the picture. (b) An EDSAC simulator developed for the Macintosh computer. The circular display (top left) shows the main memory monitor tube. The register panel (bottom left) displays the accumulator and other registers. The clock (middle right) gives the elapsed EDSAC time. The output panel (top right) shows the teleprinter printout. The central panel contains the five user-accessible controls: Clear, Start, Reset, Stop and Single E.P. The dial (bottom right), added in 1951, enabled a single decimal digit to be entered into the machine.

Technology and architecture

Before we discuss programming for the EDSAC, however, we need to know something of its technology and architecture. Figure 1a is a photograph of the EDSAC taken shortly after its completion in May 1949. Such has been the rate of progress in computing in the last decade that it is now possible to simulate the EDSAC on any modest 16-bit personal computer. Figure 1b shows an EDSAC "workstation" for the Macintosh computer. The functional descrip-

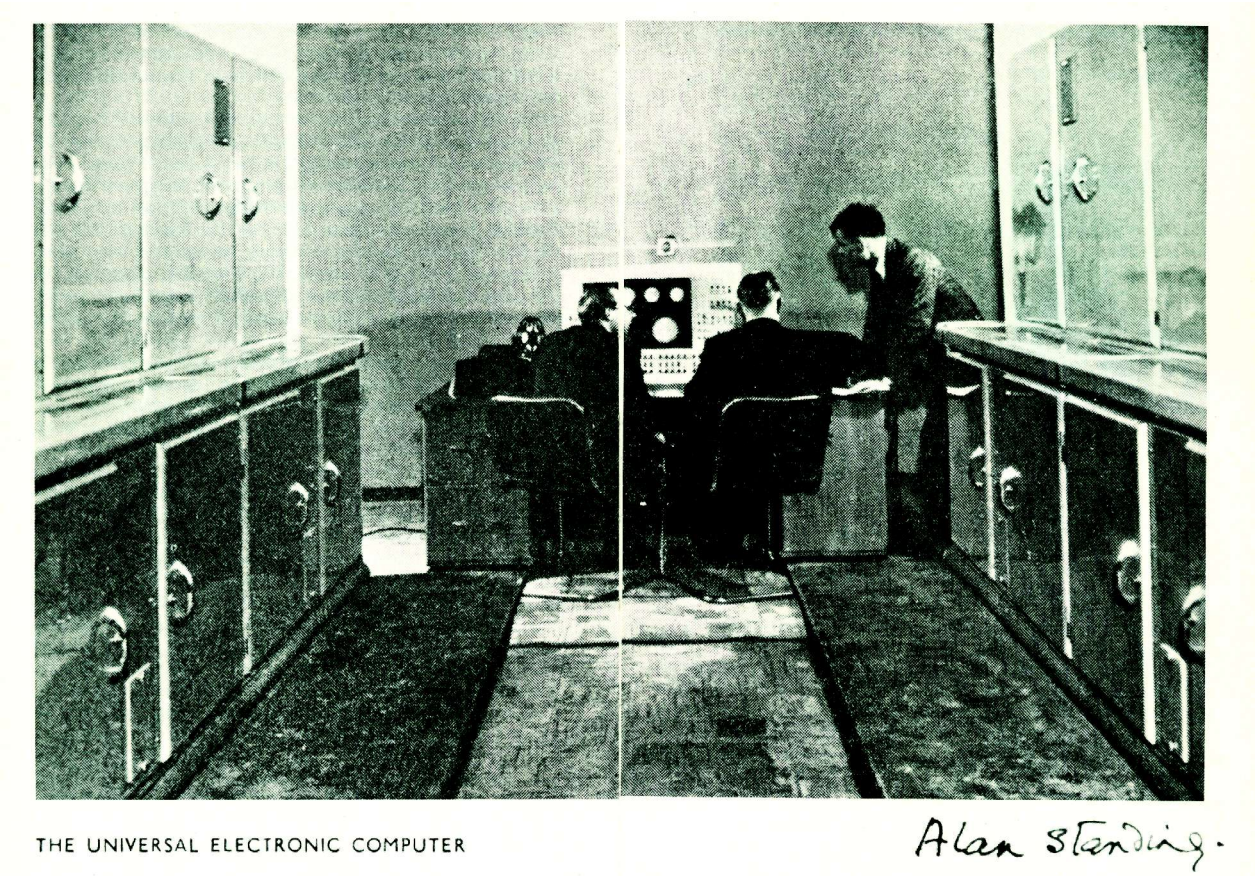
Im Unterschied zu Druckern, den bis dahin einzigen Output-Organen, arbeitet das System aus *storing tube* und *viewing tube* in Echtzeit. Hatte Cicely Popplewell, eine der beiden Programmier-Assistinnen (!) Turings (!), die Maschine eingeschaltet, sah sie also folgendes: "A bright band on the monitor tube indicated that the waiting loop had been entered." Dann rennt sie einen Stock höher, um den Lochstreifen mit dem Programm in

den *tape reader* zu stecken, und rennt dann wieder runter in den "Maschinenraum".[34] Wenn alles gut geht: "Immediatly the spots on the display tube entered a mad dance."[35] Dieser Tanz sei, so Williams, am Anfang nur der Totentanz des Programms gewesen, das ohne Fehlermeldung abstürzte. "But one day it stopped and there, shining brightly in the expected place, was the expected answer. It was a moment to remember."[36]

An dem mit 40 Williams Tubes arbeitenden Rechner in John v. Neumanns *Institute for Advanced Studies, IAS,* wird die Möglichkeit, über eine *viewing tube* den Inhalt einer beliebigen Speicherröhre - oder "den Unterschied zwischen zwei beliebigen Speicherröhren" - sichtbar zu machen, vor allem zu Beobachtung von Iterationen, etwa eines metereologischen Programms verwendet, und dazu, Alternativen des laufenden Programms gleich auszuprobieren.[37] In Manchester sorgen einige Tricks dafür, ein Programm zur Sichtbarkeit verlangsamt oder step by step durchlaufen zu lassen.[38] Fehlersuche in Programmen nach diesem Verfahren scheint schnell beliebt zu werden und heißt in Manchester *peeping*. Programmiererinnen und Programmierer um 1950 beginnen, so Reifenrath, die Lektüre von Mustern, also "graphische Kontrolle" der Auswertung ausgedruckter Zahlenkolonnen vorzuziehen.[39]

Was „Alan standing“ auf dem bekannten Photo, gestützt auf den Tisch von Ferranti Mark I gesehen hat, dürfte also genau das gewesen sein: zwei Röhren, die ein Bild vom Zustand zweier Speicherröhren geben, und oben vier kleinere Röhren, die vier Register anzeigen.

[34] Zit. bei Campbell-Kelly, 1980: 135, aus: C.M. Popplewell, A user's view of the 1949 Manchester machine, Unpublished notes, 1969.
[35] Williams, 1974: [4].
[36] Ebd..
[37] Goldstine, 1972/1993: 317.
[38] Vgl. vor allem: Martin-Campbell-Kelly, The Airy Tape: An Early Chapter in the History of Debugging, in: IEEE Annals of the History of Computing, Vol. 14, No. 2, 1992, S. 16 – 26.
[39] Reifenrath, 1998: 53. - In der Anordnung des Ferranti Mark I sind es die Bitmuster von zwei Röhren und drei Registern, die auf dem Display erscheinen.



Nebenbei aber wäre die Nullversion eines elektronischen Computerbilds entstanden: ein Testbild[40], das nichts als der Zustand der Maschine selbst ist. Es wäre von Anfang an - nicht nur als Kriegsgeschichte der Maus[41] - eine interaktive Oberfläche: als Hilfstechnik für "Debugging"[42], die Kommunikation des Programmierers mit programmierbaren Maschinen. Der Bildschirm, das "Datensichtgerät", simulierte dann zunächst nicht Welt überhaupt, sondern wäre Bild und Darstellung der Welt der Maschine, einer Welt binärer Symbole.[43] Erst als Williams Spieltrieb für seinen Artikel den Schriftzug „CRT“ aus hellen Punkten auf die Röhre zaubert, also ein Programm schreibt, das etwas anderes darstellt als es selbst ist, das heißt genauer und zeichentheoretischer: das, was es selbst ist, nur noch „bedeutet“, nämlich eine CRT, eine *Cathode Ray Tube*, aber programmtechnisch völlig sinnlos ist – erst in diesem Moment findet ein erster Umschlag ins Bild statt.

[40] Vgl. auch BOTTOM-UP. Testbild-Austellung IfT Kunst+Medien e.V., Insitut für Theoriedesign (www.theorie.com).
[41] Vgl. Roch 1995.
[42] Maurice Vicent Wilkes vom Computer Departement in Cambridge: "It was on one of my journeys between the EDSAC room and the punching equipement tehat 'hesitatin at the anglees of stairs' the realization came over me with full force that a good part of the remainder of my life was going to be spent in findeing errors in my own programs." (zit. bei Campbell-Kelly, 1992: 22, ders, 1980: 24).
[43] Vgl. F.A. Kittler, Die Welt des Symbolischen, eine Welt der Maschinen, in: Ders., Draculas Vermächtnis. Technische Schriften, Leipzig 1993, S. 58 – 80.

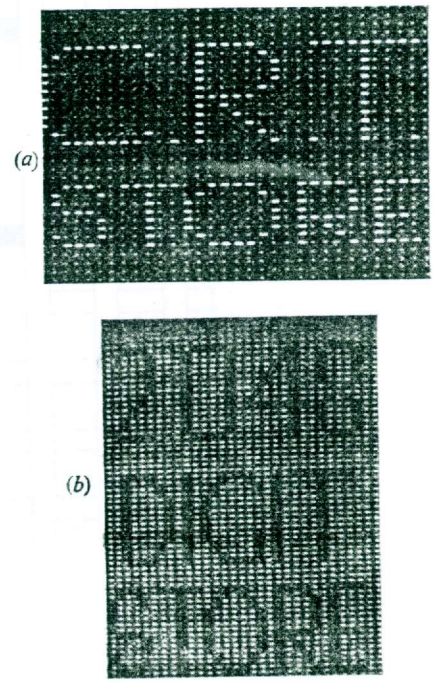


Fig. 2.—C.R.T. displays.
(a) 1 024 digits.
(b) 2 048 digits.

1.4 Resumée: Speicherpunkte - Bildpunkte

Kein anderes Zahlensystem als das binäre gibt sich in Mustern zu erkennen. Eine Zahl als "bit-Muster", ein Speicher als "bit-map", als Karte von 0/1-Mustern: binäre Zahlen, formatiert in *lines* alias *words*, adressieren digitale Strukturen in Zeilen und Spalten schon bevor sie als Zeilen- und Spalten-Bilder auf einem Rasterbildschirm erscheinen. Die "algebraischen Muster", die, nach Ada Lovelace, Babbages Analytical Machine webt, kommen zwischen Muster und *map*, Code und Bild des binären Zahlensystems zu sich. Zwar wird nicht wie in der Weberei ein Drunter und Drüber zu Mustern verwebt. Aber nur dasjenige Zahlensystem, das von der Zahl nichts als den reinen Stellenwert übrig läßt, ist als Muster lesbar. Allein darum können Williams Röhren digitale Ziffern speichern und darstellen.

Welche Rolle allerdings einzeln adressierbare Speicherelemente für eine Geschichte einzeln adressierbarer Bildelemente, also eine Geschichte von Zeilen- und Spaltenbildern spielen, steht noch dahin. So hat trotz aller Interaktivität kein digitales Bild auf der screen eines Monitors das Problem, den elektrostatischen oder quantenphysikalischen Wert

eines Pixels vom Bildschirm wieder auszulesen. Routinen, die Befehle wie *get_pixel* ausführen, lesen den Video-Speicher und nicht das Video-Signal, das bei Williams selbst zum Träger binärer Information wird. Computerbildschirme haben einen binär codierten Input, aber keinen anderen Output als Licht und Lumineszenz in die Augen der Benutzer. Und die Maus tastet bekanntlich keine Lichtwerte ab, sondern die zwei kleinen stroboskopischen Lichtschranken für x- und y-Werte, die die drei Freiheiten des Gummiballs in ihrem (aufschraubbaren!) Gehäuse in zwei rückübersetzt, ordnen räumliche Koordinaten vor dem Bildschirm: auf der Fläche des diesseits aller desktops mouse-vermessenen alten Schreibtisch, den Koordinaten eines Video-Speichers zu.

Doch Williams Röhren zeigen, unter welch anderen Vorzeichen als in Weberei und Bildtelegraphie Code und Bild, Bilder aus adressierbaren Punkten, Zeilen und Spalten in den Raum des elektronischen Bilds treten.

2.0 Display Systeme

Ein System, in dem Speicher, die der Zustand des Systems sind, und Displays, die den Zustand des Systems anzeigen, der gleichen Röhrentechnik folgen, mag technisch gesehen zum ersten Mal Punktgenauigkeit als Problem in elektronische Bilder einführen. Es mag das computergenerierte Zeilen- und Spaltenbild historisch in den Horziont stellen, der es aller prinzipiellen Analogien zum Trotz von Weberei und Bildtelegraphie unterscheidet. Williams' Röhren aber sind noch längst nicht das, was aus bildtheoretisch zufälligen Punktmustern digitale Bilder macht: sie sind kein *Display System*, also ein System, das über den Code von Bitmustern die Erzeugung eines Bildes steuert statt zeilen- und spaltenweise geordnete Speicherstellen nur anzuzeigen.

Eine Geschichte texturaler Bilder, die mit der Weberei beginnt, geht von der Hardware rasternder Techniken aus. Auch jene "Texturen", die in der modernen Graphikprogrammierung Oberflächenstrukturen - Metall, Sand, Dinosauerierhaut - simulieren und für deren Programmierung das TEXEL als kleinstes Bildelement das PIXEL zu ersetzen beginnt, sitzen rasternder Hardware auf. Bildtexturen als Hardware-Geschichte können also für einen Augenblick Reflexionsmodelle und deren Implementierung hintanstellen[44] und vom Begriff eines digitalen *Display Systems*

[44] Wie sehr schon das Wirken der erwähnten beiden Programmiererinnen Turings, Cicely Popplewell und Audrey Bate, diese Trennung Lügen straft, bleibe dahin gestellt: "Popplewell did work on ray tracing, Bates did work on symbolic

ausgehen.

Unter PC-Bedingungen, die dem Folgenden allein zugrunde liegen, besteht die prinzipielle Anordnung eines *Display Systems* aus vier Elementen: erstens dem Prozessor des Computers, zweitens einem Speicherbereich, reserviert für Graphik, drittens einem Controller (oder einer ganzen Architektur von Controllern) und viertens einem Bildschirm. Basis, Maß und Währung des Systems ist ein dimensionales Zeitwesen, das *Pixel* heißt[45]: Produkt einer akronymen, phonetischen Irritation von *piktsch*, wie *picture element*, zu *piks*, wie *pixilated,* was englisch ist und so viel heißt wie *verdreht, irriert*. Das Pixel aber, um Irritation und Rauschen auszuschließen, stellt eine Frage, die jedes digitale Bild ihm mehr als hundert Millionen mal in der Sekunde stellt: Wie ist es zu adressieren?

2.1 Bildschirmzeit

Die Zeit des Pixels wird zuerst von seinem physikalisch-technischen Ende gesetzt: dem Rasterbildschirm, jener *television-type scan line technology*, die schon Williams kennt. In der Fernseh- und Video-Technik ist bekanntlich jedes, auch ein stehendes Bild ein bewegtes Bild. Es muß soundso oft in der Sekunde von neuem "aufgebaut" werden. Die Zeilen- und Spaltenbilder des Computers sind also nicht nur in räumliche Einzelelemente zerlegt, sondern auch in zeitliche. Pixel, das ist ein Raum- und ein Zeitpunkt.

Rasterbildschirme, so ihre einfachste Beschreibung, "tasten eine Matrix mit einem Elektronenstrahl ab".[46] Der Elektronenstrahl einer klassischen Braunschen Kathodenstrahl-Röhre wandert dabei nicht kontinuierlich, sondern springt, durch einen "Sägezahngenerator" stroboskopisch diskretisiert, von Punkt zu Punkt, von Zeile zu Zeile. Was Handbücher zu "wandern" und "abtasten" verklären, ist physikalisch-technisch Krieg. In der *Cathode Ray Tube* (dem einzigen hier zugrunde liegenden Standard) trifft auf die mit Phosphor beschichtete Fläche vor (von der Maschine aus) oder hinter (vom Benutzer aus) dem schützenden Glas des Bildschirms ein "Bombardement von Hochgeschwindigkeits-Elektronen". Es erzeugt dort das quantenphysikalische Phänomen der *Kathodolumineszenz*.[47]

---

logic for her Master's thesis ... " (Campbell-Kelly, 1980: 136) !

[45] Das Hamburger Projekt *P-I-X-E-L* beschäftigt sich seit Jahren mit der Theorie der Pixelbilder. Vgl. etwa Claudia Reiche, PIXEL. Erfahrungen mit den Bildelementen. Achtung: Neue Probleme beim digitalen Signalprocessing!, in: Science & Fiction, Rundbrief, Frauen in der Literaturwissenschaft, Nr. 48, August 1996, S. 59 - 64).

[46] Schaum: 35.

[47] Vgl. Sherr, 1993: 89. Im Unterschied zur "Elektrolumineszenz" aller schönen Leuchtdioden etwa, bei der die

Bei Beschuß durch Elektronen springen die Phosphoratome in quantenphysikalisch höhere Anregungszustände und geben dabei einen bestimmten Teil ihrer Strahlung als Licht ab. Kehren sie in den niedrigeren Anregungszustand zurück, emittieren sie ebenfalls Licht. Der erste Effekt heißt *Fluoreszenz*, der zweite, stärkere *Phosphoreszenz*.[48] Die Zeit dieser leuchtenden Übergänge aber ist kurz, je nach *Persistenz* des Phosphor-Typs in den weiten Grenzen von 600 Millisekunden (Phosphor P-28) bis 0.12 Mikrosekunden (Phosphor P-16).[49] Der Elektronenbeschuß muß darum schnell wiederholt werden, wenn ein flimmerfreies Bild entstehen soll. Die *critical fusion frequency*, *CFF*, das Maß der Flimmerheit als Integration kinematogaphisch-physiologischer Größen und quantenphysikalischer, gibt das erste Zeitformat jedes Display-Systems: Wie oft in der Sekunde durchläuft die Bildschirmröhre jeden Punkt, jede Zeile, das ganze Bild: "wie oft wird das Bild in der Sekunde neu gezeichnet"?[50] Der Jargon nennt dieses Format *refresh*-Rate.

Für eine Geschichte des Zeilen- und Spaltenbilds geben bereits diese physikalischen Parameter der Bildschirmtechnik einen theoretisch neuen Ausgangspunkt: Zeilen und Spalten sind als Zeilen und Spalten in der Zeit zu denken. Wo in der Weberei die Zeit der Vorbereitung und Verarbeitung zählt: Einlesen oder Stanzen, Abtasten und Transport von Lochkarten, wo in der Bildtelegraphie die Zeit der Übertragung zählt und kostet, da zählt in digitaler Rastergraphik die Zeit des sekündlichen Bildaufbaus selbst.

Das Zeitbild der Phosphore und Elektronenstrahlen aber ek-sistiert nur in der numerischen Logik von Zeilen und Spalten. Wer etwa unter LINUX das eingangs erwähnte Graphiksystem XWindows konfiguiert, muß drei einfache Frequenz-Größen kennen: erstens wie oft in der Sekunde der Elektronenstrahl, der das Bild erzeugt, an- und ausgehen kann, genannt die *Bandbreite* und gerechnet in Megahertz (MHz); zweitens die Anzahl der Zeilen der Bildschirm-Matrix, die der Strahl in der Sekunde schreibt, genannt die *horizontale Scanrate* und gerechnet in Kilohertz (KHz); und drittens die filmische Größe, die Anzahl vollständig geschriebener Bilder pro Sekunde, genannt die *vertikale Scanrate* und gerechnet in Hertz (Hz). Drei Frequenzen also, die drei Matrizendimensionen sind.

Ein besserer Bildschirm schaltet den Strahl mit einer Frequenz von bis zu 135 MHz, also

Lichtemission durch Veränderung elektromagntischer Felder entsteht und nicht durch Beschuß.
[48] Foley, 1994: 158, und Sherr, 1993: 89 ff..
[49] Vgl. Sherr, 1993: 91.
[50] Foley, 1994: 159.

135 Millionen Mal in der Sekunde EIN und AUS. Er produziert also potentiell 135 000 000 Pixel in der Sekunde. Horizontale und vertikale Scanraten im Rahmen dieser Bandbreite hängen dann von der Anzahl der Punkte in einer Zeile der Bildschirmmatrix ab und von der Anzahl der Zeilen in dieser Matrix. Das heißt gerechnet: Um etwa ein Bild mit 800x600 Pixeln bei flimmerfreien 85 Bildern pro Sekunde zu betreiben, werden 40,8 Millionen Pixel und 51000 Zeilen in der Sekunde geschrieben.[51] Macht für die flüchtige Zeit eines Pixels 1/40800000 s = 24^-9 s = 0,000000024 s = 24 Milliardelstel Sekunden, das ist: 24 Nanosekunden. -

Der rasende Elektronenstrahl, der den Takt solcher Bildzeiten gibt, bewegt sich Zeile für Zeile schreibend und feuernd von links nach rechts und springt dann zum Anfang der nächsten Zeile zurück - ohne zu schreiben. Er wird am Ende der Zeile ausgeschaltet. Der Elektronenstrahl folgt also der textural-textuellen Technik einer Zeilenschrift, die in der griechischen Schriftentwicklung einmal die Linie des Bustrophedon - von links nach rechts, Kehre, von rechts nach links, Kehre, usw. - ablöste, in der "die Ökonomie des Schreibers mit der Landmanns", der den Ochsenpflug zieht, konvergierte.[52] (Der Grund aber für den Zeilensprung im elektronischen Geviert des Bildschirms dürfte, im Unterschied zu Derridas Lehre, wieder nur Ökonomie sein: Nach dem Modell des Bustrostrophedon müßte nach jeder Zeile ein Dekrementierungsbefehl den Inkrementierungsbefehl ersetzen, mit Zeitverlusten höher als die Rücklaufzeit.)

Der Elektronenstrahl muß jedoch am Ende jeder Zeile nicht nur ausgeschaltet werden, die Steuerfrequenz, die ihn von links nach rechts ablenkt, muß auch abgebremst werden. Er läuft darum ein Stück weiter, bevor er an den Anfang der nächsten Zeile springt. Jedes digital normierte Bild ist darum gerahmt von einer Zone der Abweichung, des Einschwingens. Der Zeitpunkt des Rücklaufs aber, des Schreibens einer neuen Zeile, antwortet exakt auf ein bekanntes Problem, das von der Bildtelegraphie bis zum TV alle übertragenen Bilder kennen: Synchronisation. Im digitalen Display-System geben ein horizontaler oder vertikaler "Synchronsationsimpuls", kurz: *Horizontal* oder *Vertical Sync*, zum bestimmten Zeitpunkt den Befehl zum Rücklauf des abgeschalteten Strahls. Sie bewirken die sogenannte horziontale und vertikale "Dunkelsteuerung".

Alle Zahlen nun der Zeiten, Zeilen, Spalten eines digitalen Rasterbilds sind, im Unterschied zum Fernsehen, veränderbare Zahlen. Sie machen das Rasterbild erstens,

[51] Also eine horizontale Frequenz von 51 KHz und eine Pixelfrequenz von 40,8 MHz.
[52] Vgl. Jacques Derrida, Grammatologie, Frankfurt a.M. 1983: 494, und Cornelia Vismann, Starting From Scratch: Concepts of Order in No Man's Land. In: War Violence and the Modern Condition (ed. by Bernd Hüppauf), Berlin, New York, 1997: 46 - 64, 47 f..

wie jedes Zeilen- und Spaltenbild seit der Weberei, zum abzählbaren Bild und zweitens sein Format, wie kein anderes Zeilen- und Spaltenbild zuvor, zu Software, also konfigurierbar. Da weder eine begrenzte Anzahl von Kettfäden noch die Breite einer Lochkarte oder die einer Zeile im Zwischenklischee die Zahl der Zeilen und Spalten des Computerbilds ein für alle Mal festlegen, darum ist die erste programmierbare Tatsache programmierbarer Bilder ihr Format, im Jargon: ihr "Modus". Der "Textmodus" ist ein einziger und relativ konstanter Standard; jeder "Graphikmodus" steht in Konkurrenz zu hunderten anderen, deren Urwald schier unüberblickbar ist.

Schreiben und Lesen von Formatinformationen laufen, nicht anders als die Operationen im Prozessor des PC, über jene Spalten- und Zeilen-Zauberei, die dem Stauferkönig Friedrich II. nicht weniger schuldet als John von Neumann: Registerführung.[53] Sie laufen also über kurze 8, 16, 32 bit lange Speicherzeilen, auf denen und mit denen ein Prozessor prozessiert. Die Frage ist nur, wo in einem Display System sitzen die Register, die Formatinformationen aufnehmen? Die Register des Prozessors sind heilig und der Bildschirm trotz seiner physikalisch-elektronischen Komplexität ist ein "'dummes' Gerät"[54], beispielsweise "blind gegen die horizontale Dunkelsteuerung".[55] Der Elektronenstrahl, das zum Leuchten bestellte Herz des Bildschirms, wird also von außen gesteuert, von einer Steuereinheit, dem *Controller*. Er ist jene programmierbare oder zumindest einstellbare Zwischeninstanz, die über den dummen Bildschirm herrscht. Der sinnige Effekt: Der Controller kann den Bildschirm, durch Überreizung der horziontalen Scanrate etwa, problemlos auch zerstören. Die Intelligenz des über eine Portnummer und die Assemblerbefehle IN/OUT vom PC-Prozessor adressierbaren Controllers besteht aus einer Architektur beschreibbarer, auslesbarer und miteinander verwobener Register. So steht im ersten der 25 acht bit breiten Register, dem Register 00h, die Anzahl der Bildpunkte pro Bildschirmzeile; in Register 02h das Pixel, nach dem der Elektronenstrahl ausgeschaltet wird und bis zum Rücklauf im Dunkeln weiterläuft; im Register 04h steht das Pixel, nach dem der horizontale Rücklauf des Elektronenstrahls beginnt; in Register 06h die Gesamtzahl der durchlaufenen Zeilen; da diese in der Regel mehr als 2^8 = 256 ausmachen wird, stehen ein neuntes und zehntes bit in einem anderen Register, usw., usw..[56] Alle pixelbezogenen Angaben werden durch 8 geteilt, denn Einheit, Währung, Sprache in der Kommuniktion mit dem Controller ist ein Zeichen des Textmodus und ein Zeichen ist 8 bit breit: der PC, die Büromaschine, das Schreibgerät. Die Manipulation dieser Register ist direkt, über Port- und Registeradressierung oder über die fertigen Modi

[53] Vgl. Cornelia Vismann, Akten. Medientechnik und Recht, Frankfurt a.M. 2000: 134 - 154.
[54] Tischer, 1988: 582.
[55] Schaum.
[56] Vgl. Tischer, 1995: CD-ROM, 9.8.10: Die Register der EGA- und VGA-Karte, CRT-Controller.

eines *Basic Input/Output-Systems BIOS* möglich. Jenseits des PC verwendet etwa LINUX die Konfigurationsdatei "XF86Config" mit Angabe sogenannter Modelines, das sind: Code-Zeilen, die die Zeilen und Spalten des entstehenden Graphikmodus festlegen (alle Angaben hier in der Einheit Pixel).

```
# 800x600 @ 85 Hz, 55.84 kHz hsync
Modeline  "800x600"    60.75  800  864  928 1088   600  616  621  657 -HSync -VSync

# 640x480 @ 85 Hz, 43.27 kHz hsync
Modeline "640x400"     36     640  696  752  832   480  481  484  509 -HSync -VSync
```
[57]

Ein digitales Computerbilder, das ist zunächst nicht mehr als die erfolgreiche Kommunikation von Prozessor, Controller, Bildschirm[58] über das Format des zu generierenden Bilds, das ist: die Beschreibung der Matrix, die das Bild erzeugt.

2.2 Adressieren: Speicher-Dimensionen

Alle Formatierungen in den Registern des Controllers haben einen Referenten: die Bilddaten im Video- oder Graphik-Speicher, jenem vierten Element eines Display Systems. Der für Bildinformationen reservierte Speicher im PC ist erstens begrenzt, je nach Graphikkarte und Videomodus auf den Teil eines Teils im RAM, der von der Speicheradresse A0000h bis zur Adresse BFFFFh geht; historische Standards wie EGA (Enhanced Graphics Adapter) und VGA (Video Graphics Adapter) waren zunächst auf genau 64 dieser 128 Kilobyte beschränkt (von A0000h bis AFFFFh). Zweitens steht die gesamte Bildinformation eines digitalen Speichers aufgereiht auf einem einzigen eindimensionalen Faden von bits. Nur weil das Format - horizontale, vertikale Synchronisationsimpulse und die interne Organisation des Speichers - den Faden in die zweite oder dritte Dimension verkreuzen, wird die schier endlose Reihe der bits überhaupt als Bild lesbar. Digitale Computerbilder und ihr Grundelement, das Pixel, sind von Anfang an dimensionale Gebilde: von der eindimensionalen Serie zu Zeilen und

[57] Die Zeile Modeline für den Modus: 800 Pixel alias Spalten auf 600 Zeilen, liest sich dann so: 60.75 ist die Pixelfrequenz oder Bandbreite des Modus, heißt: 60.75 Millionen Pixel in der Sekunde; 800 die Anzahl der sichtbaren Pixel pro Zeile oder das Pixel, ab dem der Elektronenstrahl für die horizontale Dunkelsteuerung ausgeschaltet wird; 864 das Pixel, ab dem der Strahl zum Anfang zurückläuft; 928 das Pixel, bei dem er - immer noch ausgeschaltet - die neue Zeile beginnt; und 1088 das Pixel, ab dem der Strahl wieder angeschaltet wird und, hier bei 1 beginnend, einen neuen Zyklus zum Schreiben einer Zeile beginnt. Die vier Zahlen ab 600 geben die gleichen Werte für die vertikale Steuerung des Strahls an. Die Bandbreite von 60.75 MHz errechnet sich dann nicht aus 800x600x85, sondern aus 1088x657x85.

[58] Vom Programmierer mitunter ebenso euphorisch bejubelt wie Williams' Tanz der *spots*.

Spalten zu Ebenen aus Zeilen und Spalten.

Die interne Speicherorganisation besteht fürs erste einfach in der Anzahl der bits, die auf ein Pixel fallen. Also in der eindeutigen Zuordnung von bit und Pixel, memory und screen als Grundbedingung von *bitmapped graphics*. Die bildtelegraphische Frage, wieviel Information einem Bildpunkt aufgeladen wird, kehrt hier zurück. Das Pixel digitaler Bilder aber trägt nur eine einzige Information: die Menge an darstellbaren Farben, dimensional gesprochen: die "Farbtiefe". Mit 1 Pixel = 1 bit wären genau zwei Farben darstellbar, mit 4 bit pro Pixel 16 Farben, mit 8 bit 256, mit 16 bit 65536, mit 24 bit 16777216 Farben. Alles entscheidet sich nun daran, wie Pixel-Adressen und bit-Ordnung zusammenspielen. Frühe, prähistorische Standards vom Ende der 80er Jahre des 20. Jahrhunderts, die ganze 2 hoch 4 = 16 Farben darstellen, reservieren für ein Pixel ein bit und bauen dann die Farbinformation in die dritte Dimension: vier sogenannte *bit-planes*. Das heißt: Von einem bit alias Pixel aus ist das jeweils entsprechende bit der drei anderen Ebenen zu manipulieren, zu setzen, nicht zu setzen, zu lesen, nicht zu lesen. In solchen texturalen Bildstrukturen liegen also Ebenen aufeinander oder hintereinander wie die Seiten in der dritten Dimension von Texten.

Der verschlungene Zeilen-und-Spalten-Zauber einer EGA-Routine zum Setzen eines Pixels besteht dann darin, aus der Angabe des x-, also Zeilen-Werts, und des y-, also Spalten-Werts eines Pixels und der Formatinformation des Controllers die Adresse des bits zu berechnen, dem das Pixel hörig ist. Die elementare Raster-Arithmetik lautet dann einfach: Zeilenlänge des Formats mal Nummer der Zeile, also dem y-Wert des Pixels, minus eins plus dem x-Wert des Pixels, der nichts anderes als das Pixel in der Zeile Nummer y ist. Da Adressen in bytes und nicht bits codiert sind, also der eindimensionale Faden der bit-Informationen in Paketen zu 8 bit = 1 byte abgepackt ist, müssen die x-Werte durch 8 geteilt werden; um dann das gesuchte bit aus einem solchen Byte zu holen, wird mit dem Rest gerechnet, der bei der Division durch 8 übrigbleibt, sprich: "modulo 8"; aus dem ermittelten Wert werden über sogenannte *latch*-Register (*latch:* Klinke, Drücker, Schnappschloß)[59] die einzelnen bits der bit-planes angesprochen - usw., usw..

[59] Zur Maschinenlogik von latches oder Schnappschlössern vgl. Peter Berz, 08/15. Ein Standard des 20. Jahrhunderts, Kapitel B.IV: *Schloßmodul: Werke*, und Claude E. Shannon, Eine symbolische Analyse von Relaischaltkreisen, in: Ders., Ein/Aus. Ausgewählte Schriften zur Kommunikations- und Nachrichtentheorie (hg. Kittler, Berz, Hauptmann, Roch), Berlin 2000, S. 179-216; 213- 215: *Entwurf eines elektrischen Kombinationsschlosses*. Eine C-Simulation des Schlosses, dessen Schaltplan bei Shannon auf spezielle Weise in Zeilen und Spalten angeschrieben wird, liegt vor.

Andere historische Standards vereinfachten solche aufwendigen Spalten- und Zeilenoperationen und erhöhen zugleich die Farbtiefe. Ein VGA-Modus mit 2^8 Farben etwa adressiert ein Pixel als 8 bit ohne planes; ein *Super-*, kurz: SVGA-Modus mit 2^16 = 65536 Farben als 16 bit ohne planes. Der angenehme Effekt: ein Pixel fällt entweder ganz mit seinem Adreßbyte zusammen oder mit jeweils zwei Adreßbytes. Dem glatten Aufgehen aller Adreßberechnungen in diesen Modi stellt sich jedoch ein kleines Problem entgegen: Speicherökonomie. In einem Modus, der die Bildschirm-Matrix etwa auf 800 Spalten und 600 Zeilen festlegt, ist der Speicherbedarf für jedes, auch noch so komplexe Bild 800 * 600 Pixel = 480000 Byte = 480 Kilobyte. Nun sind, wie erwähnt, für VGA-Graphik im Hauptspeicher, auf den der Controller zum ständigen refresh des Bilds und der Prozessor zu seiner Errechnung zugreifen müssen, genau die 64 KB von A0000 bis AFFFF reserviert. Um über dieses sogenannte "Speicherfenster" den viel größeren, etwa 2 MegaByte großen RAM-Speicher auf der Graphikkarte zu adressieren, muß jeweils ein bestimmter Teil der 2 MB des Karten-RAMs in das 64-KB-Fenster des Hautpspeichers "eingeblendet" werden. Der Trick ist erneut Sprung in die dritte Dimension des textural-textuellen Dispositivs. Die bits im Bildspeicher werden, statt in vier *planes*, in theoretisch beliebig viele *pages* eingeteilt, also in Portionen von 64 oder weniger KB. Die Aufgabe einer Pixelsetz-Routine ist dann, bei jeder Berechnung einer Pixeladresse die Nummer der Seite mitzuberechnen, auf der sich das Pixel befindet. Die Implementierung einer solchen elementaren Zeilen-und-Spalten-Aufgabe ist nicht mehr im textuellen Fluß der Normalsprache erläuterbar, sondern nur als zeilenweise Kommentierung einer Assembler-Routine (siehe **Anhang 1**: Listing acolset.asm).

Am inzwischen nahegerückten Horizont aller Zeilen- und Spaltenadressierung freilich steht das Ideal eines *flat memory* ohne *paging* mit 24 oder 32 bit Farbtiefe in unendlich Speicherkapazitäten. Michael Tischer 1995:

> "Der Adreßraum - unendliche Weiten. Wir schreiben das Jahr 2 hoch 32. Noch immer wird das Raumschiff Entenweiß in einem unbekannten Adreßsektor festgehalten. Tatenlos müssen wird zusehen, wie unser Speicher mehr und mehr schwindet. Die hinteren Allokatoren sind bereits zerstört, die vorderen können jeden Moment zusammenbrechen. Wir scheinen in den Saugstrahl eines riesigen Betriebssystems geraten zu sein. ... "[60]

[60] Tischer, 1995: 943. Und es sollte nicht verschwiegen werden, daß solche Poesie bei Tischer von einem Pseudo-32-bit-System, namens Windows 95 angeregt ist.

2.3 Generieren: Bresenhams Algorithmus

Routinen zum Adressieren und Manipulieren eines Bildelements - das heißt: zum Setzen einer Farbe - sind bereits der ganze Grund, auf dem ein Bild als Raster berechenbar wird. Was die Weberei auf dem Patronenpapier, was die Bildtelegraphie bei der *Autotypie* durch die Optik eines Punktrasters, beim *Zwischenklischee* durch Ablesen der Quadrate erreicht: das "Rastern" analoger Bildstrukturen, beginnt in *bitmapped graphics* mit Software: Algorithmen zur Rasterung einfachster geometrischer "Primitive" wie Linien, Kreise, Umrisse und füllende Muster in der zweiten Dimension.[61]

Linien aber in einem Raster werden nicht einfach "gezogen". Sie sind konstruktiv so wenig durch zwei Punkte und ein Lineal eindeutig angebbar wie analytisch durch ihre Steigung dx/dy. Eine Linie im Raster muß Punkt für Punkt aufgebaut werden. Doch jede Gerade, die nicht horizontal, vertikal oder diagonal ist, liegt nur ungenau im Raster. Denn jeder ihrer Punkte alias Pixel folgt dem "Alles-oder-Nichts-Prinzip"[62] und jedes Pixel ist ein Punkt der Entscheidung: Soll das Pixel über oder unter der geometrischen Linie gesetzt werden? Nur ein Algorithmus, der diese Entscheidung automatisiert, generiert und erzeugt aus zwei Werten für Anfangs- und Endpunkt eine Linie. Der Begriff einer "aktiven Matrix", den Sadie Plant aus der Weberei ableitet, kann nur algorithmisch sein.[63]

[61] Vgl. etwa Foley, 1994: *Kap. 3 Algorithmen zur Darstellung zweidimensionaler Rastergraphik.* Und Encarnacao/Straßer, 1988: 60 ff.
[62] Foley, 1994: 140.
[63] Vgl. Sadie Plant, nullen+einsen. Digitale Frauen und die Kultur der neuen Technologien, Berlin 1998.

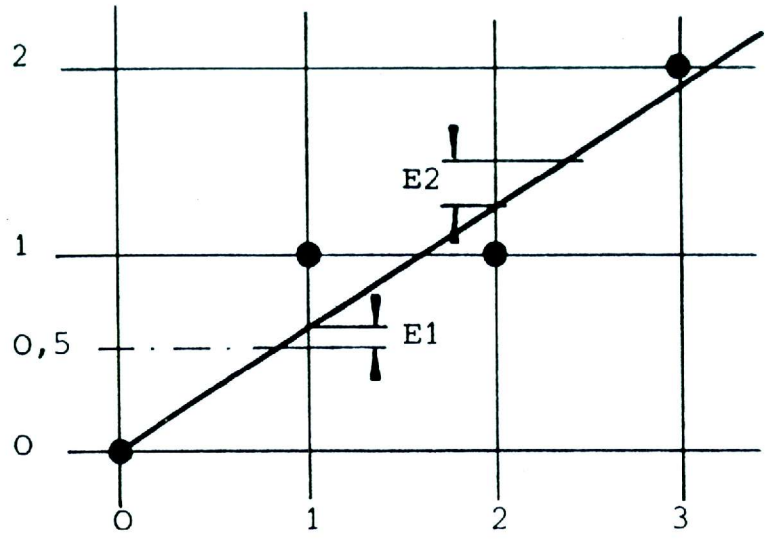


Abb. 2.28: *Erzeugung der Rasterpunkte des Vektors durch Rundung*

J.E. Bresenhams berühmter "Algorithm for Computer Control of a Digital Plotter" etwa, 1965 für IBM entwickelt, ist ein Modellfall graphisch-algorithmischer Operationen im Raster. Bresenshams Algorithmus baut inkrementell, Punkt für Punkt eine Linie auf. Eine Entscheidungsgröße *d* wird vor dem Setzen eines neuen Punkts befragt. Das Ergebnis der Befragung entscheidet, ob (wenn etwa x die Achse der Normalschritte ist) der Wert für y des nächsten Pixels gleich bleibt oder ums eins erhöht wird. Der ganze Algorithmus vermeidet jede floating-point-Division und reduziert damit schon die Berechnung einer Geraden auf das Format, in dem jedes graphische Objekt landen muß: die ganzzahligen Werte des Rasters (siehe **Anhang 2**: Bresenham.c).[64]

Jede nach Bresenham generierte Linie oder Gerade ist eine Treppe oder ein Sägezahn. Ihre Erscheinungsform, Höhe und Tiefe der Stufen, ist auch als Regelmäßigkeit eines Musters beschreibbar. Wo die Weberei bei begrenzter, zeilenweiser Adressierung durch Schäfte den Gesetzen der Bindung unterliegt, da müssen im digitalen Rasterbild analoge Elemente regelmäßige Muster aufweisen. Der Effekt heißt mit einem Begriff aus der Signaltheorie "Aliasing". Die informationstheoretische Herausforderung einer Geschichte

[64] Die vorbereitenden Schritte zu Bresenhams Algorithmus sind:
Erster Schritt: In welchem Quadranten liegt der Endpunkt?
Zweiter Schritt: Welche Relativ-Koordinate ist größer, x oder y?
Dritter Schritt: Wie müssen die Treppenschritte liegen?
Beim dritten Schritt setzt Bresenhams genialer Vorschlag ein. Da für jeden Punkt eigentlich eine floating-point-Division nötig wäre und das viel Zeit frißt, reduziert Bresenham das Problem auf einen Algorithmus, der nur Ganzzahl-Operationen und Subtraktion/Addition übrig läßt. Er führt eine Entscheidungsgröße ein, nach der die Lage des letzten Punkts die des nächsten bestimmt.

der Rasterbilder, computergeneiert oder nicht, liegt genau hier: in Whittaker-Shannons Abtasttheorem.[65] Unmathematisch gesprochen: Wenn zwei Geraden nicht nur einen Schnittpunkt haben, sondern eine ganze Gruppe von Schnittpunkten, wenn ein Punkt oder Pixel auf dem Bildschirm zwei Geraden angehören kann, dann ist aus einer gegebenen Menge von Punkten keine eindeutige Gerade zu rekonstruieren, so wie aus einem unterabgetasteten Signal keine eindeutige Kurve. Das Spiel mit dem Anderen, alias *aliasing*, ist konstitutiv für Rasterbilder.

3. Hardware - Software - Code

> In the remainder of this paper we describe our experiences. We have written it in the hope that it may speed others on toward 'Nirwana'.[66]

Spalten- und Zeilentechniken am Grund computer-erzeugter Bilder, Formate in der Adressierung von Bildelementen und elementare Verfahren zur Generierung gerasterter Bildstrukturen stellen noch nicht eine fundamentale Frage, die die Entwicklung elektronischer Rasterbilder an- und umtreibt, die Frage: Wer rastert? Ist es die Software, die rastert, oder die Hardware? Was an Bildinformationen auf Lochkarten, was an codierten Informationen bildtelegraphischer Zwischenklischees ist Software und was ist Hardware?: Das ist eine theoretische und medienhistorische Frage. Die Frage aber, ob ein C-(oder Assembler)-*Programm* die Adresse eines Pixels, den nächsten Punkt einer Linie berechnet, oder ob *Bausteine* auf der Graphikkarte dasselbe tun, ist eine praktische Frage. Genauer: Sie ist taktisch, so taktisch wie nach Spengler alle Technik.[67]

Denn die Elemente jedes Display Systems führen gegeneinander einen Krieg, genauer: einen Zeitkrieg.[68] Wenn nach den oben gegebenen Modellzahlen die vom Controller gesteuerte Bildschirmröhre 85 mal in der Sekunde auf jedes einzelne Pixel feuert, bleiben, da es nur einen Elektronenstrahl gibt, für jedes Pixel nur die berechneten 24 Nanosekunden (in Wirklichkeit noch wesentlich weniger, denn es muß die Zeit der Über- und Rückläufe mit eingerechnet werden). Theoretisch müßte also der Controller, der

---

[65] Vgl. etwa Encarnacao/Straßer 1988: 333 - 371, *Kapitel 7. Bildverarbeitung.*
[66] Myer/Sutherland, 1968: 411.
[67] Vgl. Oswald Spengler, Der Mensch und die Technik. Beitrag zu einer Philosophie des Lebens, München 1931,
[68] Für den folgenden Abschnitt vgl. Foley, 1994: 166 ff..

seine Informationen aus dem Video-Speicher bezieht, für jedes Pixel in jedem Refresh-Zyklus einmal auf den Speicher zugreifen. Nun ist die für einen solchen Zugriff nötige technische Zeit aber mindestens 80 Nanosekunden. Man sieht ein Problem entstehen. Es wächst sich aus, wenn mitgerechnet wird, daß während der Zeit des Controller-Zugriffs dem Prozessor ein ebensolcher Zugriff verwehrt ist, da es nur eine Leitung zum Prozessor gibt. Folge: Die Errrechnung eines neuen Bilds stockt. "If an attempt is made to compute concurrently with display (!), the display may develop an objectionable flicker."[69] Würde nun der Controller nicht die Information für jedes Pixel einzeln auslesen, sondern immer für 16 Pixel, sie dann zwischenspeichern (*buffering*) und schließlich einzeln alle 24 Nanosekunden zum Bildschirm schieben oder *shiften*[70], so würde der Controller 16 * 24 ns benötigen, um dieses Paket in Bildschirmpunkte umzuwandeln. Während dieser 16 * 24 ns könnte und kann der Prozessor ungestört vom Speicher lesen und in ihn schreiben.

Das grundsätzliche Problem ist damit allerdings nicht gelöst, sondern allererst aufgerissen. Denn nicht nur blockiert jeder refresh-Zugriff des Controllers den Speicher. Jede Adreßberechnung eines Pixels oder schon die einfachste Generierung einer Linie besetzen Rechenzeit und setzen den Prozessor für höhere mathematische Operationen schachmatt. Adreß- und Rasteralgorithmen arbeiten mit den Registern der *Central Processing Unit CPU*. Bereits im Juni 1968 beschreiben darum zwei amerikanische Ingenieure aus Cambridge/Mass., T.H. Myer und der bekannte Papst der Computergraphik, Ivan E. Sutherland, den Kampf um die Rechenzeit des Prozessors als ginge es um die Machtverteilung zwischen Kommandozentralen und Straßen. Sie entdecken dabei ein Gesetz, das für Display Systeme zum Prozessieren von Punkten, Zeilen, Spalten eine am Ende politisch, weil wirtschaftlich brisante Frage stellt: die nach den Grenzen von Hardware und Software.

Im historisch ersten und einfachsten Fall ist ein Display System ganz an die zentralen Prozessor-Register des *parent computers* gebunden. Der x-Wert, die Spaltenzahl des Pixels steht etwa im Accumulator-Register, sein y-Wert, die Zeilenzahl, im Input/Output-Register. "A display command is then executed which results in a point flashed on the screen."[71] Da solche simplen Operationen den Prozessor sinnlos verstopfen, wird man ein wenig Geld ausgeben und für das Display einen Zusatz kaufen: einen *data channel*. Der hat, so Myer/Sutherland, zwei eigene Register, eins für Adressen und eins, das als "word-counter" fungiert. Der Datenkanal holt sich nun sukzessive und Pixel für Pixel die

[69] Myer/Sutherland, 1968: 411.
[70] Über den sogenannten *Sequencer Controller*.
[71] Myer/Sutherland, 1968: 411.

Bildinformationen aus dem Video-Speicher (dem *display file in core*), solange bis der word-counter, der vom ersten bis zum letzten Pixel durchzählt, auf Null steht. Dann startet der Prozessor, der während des ganzen Vorgangs nun freigestellt ist, den Prozeß von Neuem.[72]

Um nun, zeit- und geldökonomisch wie die beiden Ingenieure denken, das ausgegebene Geld wieder einzusparen, macht man den *word counter* überflüssig, indem ein bestimmter Code im *display file* das Ende desselben anzeigt und den Prozessor zum Neustart veranlaßt.[73] *For just a little more money* und mit dem soeben gesparten Geld, kann man den Prozessor auch von dieser Operation entlasten: ein *jump command* im Datenkanal ermöglicht den Wiederbeginn eines *refresh*-Zyklus ganz ohne Eingriff des Prozessors. Da nun viele Bilder immer wieder die gleichen *subpictures* haben, kann man *for just a little more money* dem Datenkanal eine Anordnung für Subroutinen hinzufügen und den Prozessor von der Errechnung oft wiederholter Bildteile entlasten. Eine solche Anordnung braucht ein neues Register, in dem bei Aufruf einer Subroutine die Rücksprungadresse ins Hauptprogramm steht. Es macht nicht nur einfache Subroutinen, sondern "nested subpictures to an indefinite depth" möglich.

Was hätte man nun am Ende solcher Basteleien vor sich? Einen Datenkanal, der allein zum Display gehört und der mindestens drei Register hat, zwei für die aus dem Speicher gelesenen x und y-Werte, also eine Art Accumulator-Register und ein Register, das die Befehlsadressen mitzählt, also ein *program counter* oder *instruction pointer* und vier elementare Befehle: *load, add, halt, jump*. "From here on out one's thinking about the display changes radically." Denn der zur Entlastung des zentralen Prozessors aufgerüstete Datenkanal ist längst kein bloßer Kanal mehr: er ist selbst ein Prozessor geworden. Unversehens ist ein kleinen Spezialcomputer entstanden mit einem "begrenzten und etwas ungewöhnlichen Befehlssatz", der nur einen einzigen Zweck hat: die Prozessierung von Punkten, Zeilen, Spalten.

Doch der angefangene Weg läßt sich weitergehen - *for just a little more money*, versteht sich. Bedingte Sprünge etwa, *conditional branch instructions*, würden interaktive Elemente wie *pushbuttons* schon im Display Prozessor möglich machen; ein Stapel-oder *stack*-

[72] Wir überspringen an dieser Stelle einen Schritt: Hole nicht nur ein beliebiges Pixel, sondern addiere, für Linien, immer ein Pixel dazu: *add immediate and flash.*

[73] Die grundsätzliche Erkenntnis, die dabei abfällt: ein Display ist schon darum von an allen anderen devices unterschieden, weil ein solcher spezieller Code überhaupt möglich ist. Die einzige Funktion der graphischen Daten im Speicher ist "to post a picture on the screen" und das heißt die Daten sind so uniform, daß ein spezieller Code für den *channel halt* nicht fehlinterpretiert werden kann.

System mit Befehlen, die aktuelle Zeilen-und-Spalten-Position des Elektronenstrahls abzulegen und wieder abzuholen, *push & pop*, ermöglichte es, die Hierarchie der Subroutinen im Display Prozessor überhaupt noch zu verfolgen; schließlich würde die Möglichkeit, den Subroutinen Parameter zu überstellen, die Tranparenz erhöhen, wofür zusätzlich ein eigenes Register zum Hoch- und Runterzählen der stack-Adressen nötig ist, usw., usw.. Am Ende dieses Wegs steht ein Display Prozessor mit Registern, Stack und 17 Befehlen, der kaum mehr ein Display Prozessor zu nennen ist.

> "We have built up the display channel until it is itself a general purpose processor with a display."

Wenn man nun aber bemerkt, daß dieser neue, allgemeine Prozessor die meiste Zeit ungenützt ist, könnte man auf die Idee kommen, dem Display einen Data Channel hinzuzufügen, der zunächst nur Zeilen und Spalten, usw. ... "In fact, we found the process so frustrating that we have come to call it the 'wheel of reincarnation.' We spent a long time trapped on that wheel before we finally broke free." -

Die Geschichte einer Hardware, die seit der Weberei Punkte, Zeilen und Spalten als Bilder prozessiert, landet damit, noch bevor sie als Computergraphik recht begonnen hat, bei der Frage nach Hardware und Software selbst. Auf dem *state of the art* und seit Silicon Graphics *Geometry Engine* wird eine steigende Menge graphischer Operationen auf Zeilen und Spalten und eine steigende Menge von Matrizenoperationen zur Verrechnung von Zeilen und Spalten in Hardware implementiert. Die explodierende Welt der digitalen und programmierbaren Bilder dagegen scheint die bunte Feier nicht enden wollender graphischer Software zu sein.

Myer/Sutherlands Gesetz der Reinkarnation von 1968 stellt die Frage nach Code und Bild in diesen Horizont und Turings Konzept der universalen diskreten Maschine wird zum Schicksalsrad digitaler Geschichtszeit: *special purpose* oder *general purpose*?

ANHANG 1:

```
;       Listing einer Pixelsetz-Routine 'acolset.asm'.
;       1. Spalte: Offset der Adresse eines Befehls
;               ( = Register ip/Instruction Pointer im Segement cs/Codesegment )
;       2. und 3. Spalte: Assembler-Befehl in Hexadezimal-Code
;       Rest: Assembler-Befehl in Mnemo-Code. Eine 1 vor der Zeile heißt:
;       die Befehle stammen aus einer MACRO-Datei, eine 2: MACRO-Aufruf im MACRO.
;       Ein Semicolon bezeichnet "auskommentierte" Zeichen: Kommentare zum Quellcode.


011D  87 DB 90                              ALIGN  4
0120                            _acolset PROC  FAR
                                        ein16f  0               ;  MACRO aus GRAPH.inc,
                                                                ; Parameter 0
0120  67| 8B 5C24 04      1             mov     bx, [esp+4+0]
0125  67| 8B 4424 06      1             mov     ax, [esp+6+0]
012A  03 DB               2             add     bx, bx
012C  67| 8D 0480         2             lea     ax, [4*eax+eax]
0130  02 F8               2             add     bh, al
0132  80 D4 00            1             adc     ah, 0
0135  3A 26 0000 E                      cmp  ah, bank          ; bank = public-Variable (Typ char)
                                                               ; aus bildlarg.asm
0139  74 0F                             je      SHORT noswit3  ; wenn gleich: kein bankswitch
                                        bankchange
013B  88 26 0000 E        1             mov     bank, ah
013F  BA 03CE             1             mov     dx, ETPort
0142  C0 E4 02            1             shl     ah, 2
0145  B0 09               1             mov     al, ETInd1
0147  EF                  1             out     dx, ax
0148  40                  1             inc     ax
0149  EF                  1             out     dx, ax
014A  67| 8B 4424 08      noswit3:      mov  ax, [esp+8]       ; Farbwert vom stack ins  Register ax
014F  64: 89 07                         mov     fs:[bx], ax
0152  CB                               retf
0153                            _acolset ENDP
```

ANHANG 2:

```
/*********************************************************************************************
*  C-Funktion "Line": Zieht eine Linie nach dem Bresenham-Algorithmus.                       *
*  Aus: Tischer PC Intern 5.0/1995(CD-ROM). Kommentare (PB) zwischen: /*.                    *
*  Genauer kommentiert ist der Fall, daß die Steigung < 1, also x die                        *
*  "größere Relativkoordinate" ist und die "Einheitsschritte" in                             *
*  Richtung der x-Achse laufen, was wahrnehmungsphysiologisch oder      *
*  "subjektiv besser beurteilt wird"                                                         *
*  (Encarnacao, Computer Graphics, 3. Aufl. 1988: S. 61).                                    *
*                                                                                            *
*                                                                                            *
*  Eingabe : X1, Y1 = Startkoordinate (0 - ...)                                              *
*            X2, Y2 = Endkoordinate                                                          *
*            FARBE  = Farbe der Linie                                                        *
*********************************************************************************************/

/*-- Unterfunktion, vertauscht zwei Integer-Variablen ----------------*/

void SwapInt( int *i1, int *i2 )
{
 int dummy;

 dummy = *i2;
 *i2   = *i1;
 *i1   = dummy;
}

/*-- Hauptteil der Funktion ------------------------------------------*/

void Line( int x1, int y1, int x2, int y2, COL farbe )
{                                       /* Ziehe eine Linie von x1/y1 nach x2/y2 */
                                        /* Kommentierung: vgl. ab "Über die X-Achse laufen" */
 int d, dx, dy,
     aincr, bincr,
     xincr, yincr,
     x, y;

 if ( abs(x2-x1) < abs(y2-y1) )     /* Über X- oder Y-Achse laufen? */
  {                                 /* 1. Über die Y-Achse laufen */
   if ( y1 > y2 )                   /* y1 größer y2? */
    {
     SwapInt( &x1, &x2 );           /* Ja, X1 mit X2 und     */
     SwapInt( &y1, &y2 );           /* Y1 mit Y2 vertauschen */
    }

   xincr = ( x2 > x1 ) ?  1 : -1;   /* Spalteninkrement festlegen */

   dy = y2 - y1;
   dx = abs( x2-x1 );
   d  = 2 * dx - dy;
   aincr = 2 * (dx - dy);
   bincr = 2 * dx;
   x = x1;
   y = y1;

   acolset( x, y, farbe );          /* ersten Punkt setzen */
            xsync();
   for (y=y1+1; y<= y2; ++y )       /* Line auf X-Achse durchlaufen */
```

```
   {
    if ( d >= 0 )
     {
      x += xincr;
      d += aincr;
     }
    else
     d += bincr;
    acolset(x, y, farbe);
                    xsync();
   }
  }
 else                              /* 2. Über die X-Achse laufen.
                                        Heißt: x ist die größere Relativkoordinate */
  {
   if ( x1 > x2 )                  /* x1 größer x2? */
                                   /* also wenn x2 sich in den beiden linken
                                        Quadranten befindet */
    {
     SwapInt( &x1, &x2 );          /* Ja, x1 mit x2 und     */
     SwapInt( &y1, &y2 );          /* y1 mit y2 vertauschen */
                                   /* Aufruf der Funktion SwapInt(), s.o. */
    }

   yincr = ( y2 > y1 ) ? 1 : -1;   /* Den ermittelten y-Wert rauf- oder runter-
                                        zählen ? Runterzählen, wenn y2 oberhalb
                                        von y1 liegt */
   dx = x2 - x1;                   /* Die Steigungsdifferenzen ermitteln */
   dy = abs( y2-y1 );              /* Beim y-Wert den Absolut-Wert nehmen */
   d  = 2 * dy - dx;               /* Formel zur Errechnung des Diskriminators d
                                        für den ersten y-Wert nach dem Anfang.
                                        d entscheidet, je nachdem es positiv oder
                                        negativ ist, ob y beim nächsten x-Wert um
                                        eins erhöht wird oder gleich bleibt */
   aincr = 2 * (dy - dx);          /* Diskrimator-Wert für jeweils nächstes y,
                                        falls der vorige Wert größer 0 war */
   bincr = 2 * dy;                 /* Diskrimator-Wert für jeweils nächstes y,
                                        falls der vorige Wert kleiner 0 war */
   x = x1;                         /* Übergebene Anfangswerte */
   y = y1;

   acolset(x, y, farbe);           /* Den Anfangs-Punkt setzen */
           xsync();
   for (x=x1+1; x<=x2; ++x )       /* Line auf X-Achse in "Einheitsschritten"
                                        durchlaufen */
    {
     if ( d >= 0 )                 /* Diskriminator größer oder gleich 0 ? */
      {
       y += yincr;                 /* Wenn ja: y eins hochzählen */
       d += aincr;                 /* Neuen Diskriminator setzen */
      }
     else                          /* Wenn nein: y bleibt gleich */
       d += bincr;                 /* Neuen Diskriminator setzen */
     acolset(x, y, farbe);         /* Neuen Punkt der Linie setzen */
                      xsync();
                     }
  }
}
```