1. >>

Wolfgang Hagen

„Effekt - Affekt - Computer. - Eine kleine Fussnote zu archäologischen Fragen“[1]

Meine Damen und Herren,

2. >>

In dem letzten Band “Computer als Medium”, 1994, der das langjährige DFG-Projekt abschloss, bei dem ich Wolfgang kennenlernte, resümieren die Herausgeber Bolz, Kittler und Tholen: “Mittlerweile aber gehen technische Medien überhaupt in der Universalität von Computern auf.” Die drei hatten Recht. Computer sind heute myriadenfach ubiquitär verbreitet. Oder, um es mit Bernhard Siegert zu sagen: “Computer beginnen vor unseren Augen zu verschwinden, indem sie Teil des Gewebes dieser Welt werden”.[2]

Worauf es mir in der nächsten halbe Stunde ankommt: Auch wenn, und gerade weil Computer schon im Textil der Welt ungreifbar werden, bleiben es dennoch auf eine prinzipielle Weise unfertige Geräte. Sie sind imperfekt, weil ihre Perfektion nie und unter keinen Umständen erreicht werden kann. Das war der Punkt, mit dem Alan Turing Hilberts “Entscheidungsproblem” gelöst hatte. Die Gedankenmaschine Computer, die eine logischen Architektur hat, wie sie noch heute überall verwendet wird, hatte der verschlagene 23jährige sich bekanntlich nur ausgedacht, um zu beweisen, dass ein Computer niemals beweisen kann, dass ein Computer richtig läuft.

Es ist eine negative Simulations-Paradoxie: Turings Maschine, versehen mit einem unendlichen Band und unendlichen Resourcen, sollte das menschliche Denken simulieren können, d.h. alles berechnen, was auch ein Mensch berechnen und an Problemen lösen könnte, wenn er dafür nur unendlich lange Zeit hätte; und damit also kein Sein zum Tode. Turings Simulations-Paradoxie des menschlichen Denkens liegt in dieser endlichen Unendlichkeit, die nur im

---

[1] Vortrag anlässlich der Übergabe der Festschrift an Wolfgang Ernst, 9.5.2019, Medientheater HU Berlin

[2] Siegert, Bernhard (2016): Digitalität in der Medien- und Kulturwissenschaft (a.k.a. ‚kulturwissenschaftliche Medialitätsforschung‘), https://digigeist.hypotheses.org/80 digigeist 04/07/, 242. Siegert referenziert auf Quantum-Computer-Theoretiker: Williams, Colin P. ; Clearwater, Scott H. (2000): Ultimate zero and one : computing at the quantum frontier, New York : Copernicus, 3.

prozeduralen Vollzug aufgelöst werden kann; oder mathematisch gesprochen, alles für berechenbar erklärt, was in endlicher Aufzählung berechenbar ist.

Diese Zeitparadoxie macht die Sache heute wieder hochaktuell. Wir leben in einer Welt ungezählt vernetzter Rechner, mit Zig-Milliarden von Servern in Tausenden von hoch gesicherten Data-Centern über die ganze Welt verstreut, schier unendliche, aber doch prinzipiell aufzählbare Resourcen mit schier unendlichen, aber ebenso prinzipiell aufzählbaren Anweisungs-Tabellen, sprich Software.

Strukturell und logisch, also epistemologisch, findet sich diese endliche Unendlichkeit gleichermaßen in Turing's Gedanken-Modell in "On Computing Numbers" von 1936. Ab 1945 machte der Atombomben-Mathematiker John von Neumann die weitere technologische Expansion zu seiner Sache. 74 Jahre später: Alle weltweit verschalteten Computer sind nicht überabzählbar, aber faktisch unabzählbar. Sind sie deshalb vor unseren Augen verschwunden? Nirgendwo hängt eine Karte aller physischen Internet-Kabel. Auch nicht im Netz. Hoffnungslos. Nach Turings Logik können Computer unendliche viele Computer enthalten, die wir aufzählen dürften, aber wir können es nicht: Ein zeitliches Imperfektions-Paradox, das sich nicht auflösen löst, aber von nun an vermutlich unendlich prozedieren wird. Alle diese Maschinen können alles berechnen – außer, ob eine Maschine des eigenen Typs richtig läuft, Das ist das Imperfektions-Paradox, das ebenfalls unauflösbar bleiben wird.

Ich bedanke mich für die Einladung zu dieser Rede, die mir Gelegenheit gibt, noch einmal in den Blick zu nehmen, was mich mit Wolfgang Ernst[3] verbindet und wo es mich verbunden hat.

In Zeiten affektiv hochgejazzter Artificial Intelligence, einer unvordenklichen Datenökologie, digital-diktatorischer Überwachungs-Regimes und Phantasmagorien von autonomen Autos, probabilistischen Personalprofilen und anderen monströsen Automatisierungen; unterstützt von nur scheinbar kritisch agierenden aktivistischen Metaphysikern, die aus all dem ein neues post- oder trans-humanes Menschenbild herbeirufen, das sie am Ende aber doch nur mit hundert Jahre alten monistischen Metaphysiken a la Bergson oder Whitehead

---

[3] Allerdings, wenn Du mich das nächste Mal hören wirst, wird es um die arché-lose Archäologie der Siliziumschaltungen und ihrer Epistemologie gehen, und dabei auf die affect-effect Ökosensorik abheben, die gerade dabei ist, alles glatt zu bügeln, was je zur Differenz von Natur und Kultur gedacht wurde…
Oder, wenn Du so willst: die arché des "static between the stations", wie es Cage sagte, ist im Zeitalter der digitalen Gleichschaltung, um es vorsichtig zu sagen, eskamotiert.

beschreiben, - kurz: in diesen Zeiten schwerer intellektueller Diffusionen tut ein Blick zurück not und gut, und mit ihm die Erinnerung an die prinzipielle Unfertigkeit der herrschenden Systeme, die in ihrer Propaganda stets das Gegenteil, nämlich absolute Perfektion behaupten.

Das zeitliche Computer-Imperfekt, das ich hier markiere, bitte ich nicht zu verwechseln mit dem Widerspruch, den beispielsweise Franco Berardi zwischen Cyberspace und Cybertime auszumachen versucht hat. Berardi sagt, der Cyberspace sei zwar inzwischen universell unendlich, die Menschen aber könnten keine unendliche Cyberzeit, kämen also im Cyberspace also nicht mehr mit und daraus resultiere alle Panik und das ganze semio-kapitalistische Chaos unserer Epoche. Ich glaube nicht, dass man so anthropologisch argumentieren sollte, und plädiere hier eher für epistemologische Gründlichkeit. Diese besagt, ich wiederhole es, dass bei keinem der auf der Welt laufenden Computer bewiesen werden kann, ob sein Programm richtig läuft, bis es zuende gelaufen ist, oder, wie Turing sagte, had “come to an end after a finite number of steps[4]”. Ohne diese fundamentale Chronopoiesis der digitalen Kultur, da folge ich Wolfgang Ernst, bleibt die heutige Welt unverständlich.

Schaut man sich die Geschichte dieser Masse aller existierenden Computer genauer an, und dazu die Zeitlichkeit ihrer auf Permanenz umgestellten Evolution, so wird sehr schnell ein treibender Effekt sichtbar, den ich den Affekt-Effekt des Computers nennen möchte. Es ist gleichsam ein Paradox zweiter Ordnung, das die Eigenschaft hat, in einer gewaltigen Kaskade alter und neuer Affekte, den Computer, seine Masse und mit ihm alle seine Imperfektionen unsichtbar zu machen.

Für die Frage, wie es zu dieser Massierung kam, hat der Computerhistoriker Michael Mahoney zunächst zurecht darauf hingewiesen, dass das Turing-Modell selbst immer nur “ein offenes Schema [bot] für eine potenziell unendliche Bandbreite von Anwendungen. Wie ein Computer ganz konkret aussehen sollte, hing von [...] [denen] ab, die ihn benutzen würden und ihm ein entsprechendes Design gaben”[5]. Der Affekt-Effekt des Computers besagt also auch: Es gibt keinen Computer, es gibt nur unendlich viele Computer-Designs. Computer, das waren einmal riesige Rechnersäle; und sind heute Telefone oder

---

[4] in: Turing, A. M. (1937): On Computable Numbers - With An Application To the Entscheidungsproblem, Proceedings of the London Mathematical Society Series 2 42 (1937) pp 230–265, , 247

[5] Mahoney, Michael S. (1990): The Roots Of Software Engineering, CWI Quarterly 3, no. 4 (1990): 325 – 334, ,258, 12768_MahoneyTheRootsOfSoftwar_1990

Armbanduhren, Aufzugssteuerungen, Überwachungskameras oder Herzschrittmacher, PCs, Router und Tabletts; wieder andere, zentimeter-klein um ihre Festplatten verbaut, in den Datencenter-Rags von Google. Mit anderen Worten: Die Realität des Computers hängt von demjenigen Design ab, mit welchem seine immanenten Paradoxien den Umgebungsbedingungen am besten angepasst werden können. Solche Design-Entscheidungen aber sind schon deshalb affektiv, weil jede von ihnen das Risiko trägt, nicht beweisen zu können, ob sie richtig sind. Das spezifische Design eines Computers ist unbeweisbar, weil es sich nur im Vollzug beweist, oder anders gesagt: Design-Entscheidungen sind effektive und affektive Programmier-Entscheidungen.

"Wahrnehmungen sind Bilder, die ich 'von außen' kenne, während Affektionen Bilder sind, die ich 'von innen' kenne"[6], heisst es bei Bergson und ich bitte um Vergebung, dass ich hier noch einmal Eulen nach Athen trage, in Anwesenheit von Marie Luise Angerer, angesichts Ihrer großartigen Studien zum Affekt, von denen ich so viel gelernt habe. Ich will deshalb nur kurz erwähnen, dass der Begriff der Affektion aufs lateinische affectio zurückgeht, ein Wort, das ebenfalls aus einer Prozeduralwissenschaft stammt, und zwar aus der ersten, die überhaupt je formalisiert wurde, nämlich aus der Rhetorik. Affectio ist eine Wort-Neuschöpfung von Cicero[7] als Synonym für das griechische Wort "Pathos", weil ja die Römer in Gestalt von Cicero und Quintilian es grundsätzlich ablehnten, griechische Begriffe in ihre Sprache aufzunehmen. Also sagten sie statt Pathos "affectio", und meinten damit alles das, was eine Stimmung bezeichnet, in welchen Zustand, um es mit Turing zusagen, eine Rede sein Auditorium versetzt.

Was das Pathos oder den Affekt des Computers betrifft, so gibt es eine Situation oder besser noch: eine Szene, die allerdings so aufwühlend rhetorisch ist, wie eine Rede es wohl kaum werden kann, und in der zugleich die Unfertigkeit, die Affektion und die Simulationskraft des Computers wie nirgendwo sonst auf den Punkt kommt. Das ist, wenn's ans Programmieren geht. Einen Computer programmieren, das ist, anders als es Charles Hoare einst dekretierte, eben kein purer Logos, keine pure "mathematische Aktivität, deren zuverlässige Praxis nur die entschlossene Anwendung traditionell

---

[6] Vgl. Erik Ogers Einleitung in: Bergson, Henri (1991): Materie und Gedächtnis : eine Abhandlung über die Beziehung zwischen Körper und Geist, Hamburg : Meiner, XXXI..

[7] Thesaurus Linguae Latinae VOLVMEN I, Leipzig et. al. 1900, 1176.

mathematischer Berechnungs- und Beweis erfordert”[8], wie er sagte. Im Gegenteil, Programmieren ist eher das, was Alan Kay’s Smalltalk-Konzept vorsah, nämlich ein hoch-affektives Unterfangen, das, um mit der Unfertigkeit des Computers fertig zu werden, niemals aufhört, sondern die permanente Korrektur des gerade Programmierten erzwingt, wie bei Smalltalk, im Programmablauf selbst, sozusagen als Utopie eines unaufhörlich Herumbastelns mit konkreten Objekten und logischen Mitteln. Weder Hoare’s mathematische Abstraktion, noch Kay’s emphatische Affektion sind je in Reingestalt software-sprachlich realisiert geworden. Wobei es heute, im Zeitalter des “agilen Programmierens”, ohnehin nicht mehr um das Design idealer Programmiersprachen geht, wie noch vor drei Jahrzehnten bei Hoare, Dykstra oder Kay. Programmieren ist heute, um einen aktuellen Blog auf HackerNoon zu zitieren, ‘eine [so genannte] agile Kunst, existierende Komponenten zusammenzustecken, was man am Ende auch hinkriegen könnte ohne irgendein besonderes “computer degree”’[9]. Umso mehr aber werden darum gerade im Netz herauf und herunter die “subconscious biases” diskutiert, die unbewussten Faktoren des Programmierens, eben in den Gruppenprozesse agiler Software-Entwicklung, wo es, zum Beispiel, nur einen Frauen-Anteil von maximal 8 Prozent gibt.

Lesen wir die Selbstbeschreibungen von Programmierern, von John Backus, Gerald Weinberg und Joseph Weizenbaum bis hin zu den zahllosen Blog-Einträgen auf Hackernews, Hacker Noon, Reddit oder Stackoverflow, so zeigt sich: Programmieren ist denn doch, heute wie damals, genauso kompulsiv, immer hart an der Grenze der Sucht. Nathan Ensmenger, der Computerhistoriker, hat klug hinzu gefügt: “Im Gegensatz zu guter Literatur werden selbst die besten Computerprogramme niemals von jemand anderem gelesen als vom Autor selbst.”[10] Tatsächlich: Quellcodes, das sind die autologischsten Texte, die es gibt. In eigentlichen Sinn des Wortes werden sie ja nicht einmal von den Maschinen gelesen. Da werden sie vielmehr sofort übersetzt, und das ist genau der Punkt, der, wie es Edsger Dijkstra schreibt, ihn und seine programmierenden Kollegen immer in die höchste Affektion versetzt hat: Da haben die Texter alle formalen Anforderungen der Sprache erfüllt, sie

---

[8] Hoare, C. A. R.(1989): The Mathematics of Programming, Essays in Computing Science, Heme, Hempstead, 6.

[9] The downside of being a programmer – Alex Ewerlöf

[10] in: Ensmenger, Nathan (2010): The Computer Boys Take Over Computers, Programmers, and the Politics of Technical Expertise, Cambridge:MIT , 206

haben habe alle ingenieursmäßig definierten Ziele eingebunden und nun kommt es zu dem Punkt, “wo der menschen-lesbare Text in den maschinen-lesbaren Text konvertiert wird, mit all diesen grandiosen Stellen, wo das menschliche Auge Dinge sieht, die nur ein Menschenauge sehen kann und will, und dann soll es dennoch die trügerische Gewissheit geben, dass durch diese Konversion nichts verloren geht”[11]? Glauben Sie mir, Programmieren ist auch eine furchtbare Affektion, verstörend, aufregend und pathetisch zugleich.

Schon bevor ich zu der Gruppe um Kittler, Wolfgang Ernst und all den anderen stieß, war auch ich selbst ans Programmieren geraten. Ich arbeitete damals im Radio und hatte mit computergestützten Mitteln, zweite Hälfte der 1980er Jahre, ein Popmusik-Radio bei Radio Bremen aufzubauen, ohne dass es dafür geeignete Computertools gegeben hätte. Also schrieb ich erst in Pascal, dann in C und C++ zuerst ein Musikplanungs-Programm, [immerhin nach einer Idee von Walter Bachauer, falls den noch jemand kennt, Metamusik-Kurator und Koyaanisqatsi-Vertoner], dann ein Dienstplan-Programm und schließlich ein Nachrichten-Verteil-System, das Agi-Net hiess und von 1992 bis 2006 tatsächlich am Ende auf allen 400 Arbeitsplätzen Radio Bremens lief.

Ich weiss also, was es heisst, den Affektionen des Programmierens sich auszusetzen, die von völliger Depression über schizoide Aussetzer bis zur absoluten Allmachtseuphorie reichen können. Man hat dauernd das Gefühl, es würden einem ins Hirn urplötzlich neue Synapsen gleichsam von innen hineingeschossen. Was Bergson schon früh aus seiner Teilnahme an den spiritistischer Seancen in Clermont-Ferrand erfuhr, das erfährt der Programmierende heute noch heftiger: Bilder von innen, mächtige kathedralische Architekturen, die, von außen gesehen, gar nicht existieren.[12]

Was insofern zu dieser Affektion des Programmierens auch gehört, ist die Erfahrung, dass eine bestimme Klasse von Erfahrungen nur durch sich selbst erfahren werden kann, wie es William James so eindrücklich ebenfalls aus seinen Spiritismus-Studien gezogen hat. Beide, James und Bergson, präsidierten um 1900 jeweils für eine Zeit die “Society for Psychical Reasearch”, also im Kernforschungszentrum des Spiritismus um 1900. Zur der von ihnen

---

[11] Dijkstra, Edsger W. (1976): A discipline of programming, Englewood Cliffs, NJ ; Prentice-Hall ; ,213

[12] Gewiss erlebt man dabei die ganze Kaskade der Affekte, wie sie “CICERO in <De oratore> behandelt [...], nämlich amor: (Zuneigung), odium (Haß), iracundia (Zorn), invidia (Neid), misericordia (Mitleid), spes (Hoffnung), laetitia (Freude), timor (Furcht) und molestia (Verdruß).” in: Krämer, J. (1992): Affektenlehre, in: Ueding, Gert (1992): Historisches Wörterbuch der Rhetorik - Bd. 1, A - Bib, Tübingen Niemeyer , 218-253, , 222

gefundenen Klasse der Erfahrungs-Erfahrungen gehört eben auch die Prozedur des Software-Schreibens, die, statt der mediumistischen Seance, das Medien-Apriori des Computers hat.

Dass Programmieren eine unhintergehbare Affektion entfaltet, darin war ich übrigens auch mit Friedrich Kittler einig und mit all denen, die in seinem donnerstäglichen Programmier-Seminar saßen. Programmieren versteht nur, wer es auch tut. Und wir forderten deshalb für den wahren Philologen das verpflichtende Erlernen einer Programmiersprache.

Vieles von dieser Affektion des Programmierens findet sich auch in jenem legendären Vortrag wieder, den Friedrich Kittler im Frühjahr 1991 auf Einladung von Sepp Gumbrecht in Stanford hielt, unter dem gegensinnigen Titel “There is No Software”, obwohl es doch darin heisst: “Moderne Medientechnologien sind, schon seit Film und Grammophon, grundsätzlich daraufhin angelegt, die Sinneswahrnehmungen zu unterlaufen. Wir können schlichtweg nicht mehr wissen, was unser Schreiben tut, und beim Programmieren am allerwenigsten.“[13]

“There is no Software” machte schon vor annährend 30 Jahren die Rechnung auf, mit welcher babylonischen Sprachenflut wir es zu tun kriegen würden, bei der gegebenen Geschwindigkeit der Evolution der von Neumann Computer. Für Kittler, der ebenso viel von Hölderlin verstand wie vom Assembler, war klar, dass jetzt, zum historisch ersten Mal, jenseits aller literarischen Sprachen a la Hölderlin[14], wo schon die bloße poetische Andeutung eines Blitzes einen tatsächlichen Blitz vor unsere Augen schicken kann, Sprachen auf Maschinen-Ebene des Computers laufen, getrieben von der purer Elektrizität myriadenfacher An-Aus-Schaltungen, die sich unserem Wissen entziehen, insofern wir Programmierer bestenfalls eine Wette abschließen können, dass die Sachen, die wir geschrieben haben, darauf laufen werden.

“There is no Software” beschreibt nicht nur diesen affektiven Schock der Ankunft der Herrschaft der Maschinensprachen, sondern unternimmt eine dekonstruktive Kritik an den neuen gouvernementalen Regimen der Software-Industrie. Kittler konterkariert sie am Ende mit einer hypothetischen Vision und ringt dabei sichtlich mit dem Gedanken, was es denn eigentlich bedeutet, für

---

[13] Kittler, Friedrich (1993): Es gibt keine Software, in: ders.: Draculas Vermächtnis. Technische Schriften. 225-242, Leipzig , 229.

[14] Kittler, Friedrich A. (1992): There Is No Software, Stanford literature review, 9, 81-90, 82.

das Sprachwesen Mensch, wenn nicht mehr die Sprache seine Daseinsform bestimmt, sondern seine Daseinsform im Begriffe ist bestimmt zu werden von einem Maschinen-Diskurs, der paradoxerweise nur deshalb existiert, weil die Menschen ihn zwar mit menschlicher Sprache bestenfalls noch antriggern können, aber dabei nur Unvollkommenes zuwege bringen. Und das umso mehr, als die neuen Machtbeziehungen der Prozessor- und Softwareindustrie das Ganze noch weiter verformen. Dem ganzen Vortrag spürt man an, wie Kittler mit den Unvollkommenheiten der Turing Maschine nicht wirklich leben will. Er kennt ihre Eigenschaften sehr genau, nämlich dass mit ihr nicht beweisbar ist, dass sie richtig läuft und dass sie nur berechnen kann, was unendlich, aber eben nur aufzählbar unendlich ist. Darin genau liegt für Kittler der Skandal. Wenn Turing Maschinen real werden, versehen mit begrenzten Silizium-Arbeitsspeichern und endlichen Solid State-Festplatten, kommt, um es mit Wolfgang Ernst zu sagen, ihr chronopoetisches Paradox zum Tragen und sie können in realer Technologie eben keineswegs alles berechnen, was zu berechnen wäre. Dass in einer Turing-Maschine alles Berechenbare in endliche Schritte aufgelöst, also diskretisiert werden muss, verweist in Kittlers Augen “auf eine prinzipielle Unmöglichkeit der Digitalisierung des Körpers der reellen Zahlen, also die ehedem so genannte Natur.”[15]

Man kann es auch anders sagen: Kittler sieht in der Imperfektion der Turing Maschine den Grund für die neue Macht- und Verknappungs-Regime a la IBM, DOS oder Apple. Dagegen stellt er die Vision einer Maschine, die “reine Hardware darstellen” würde und sich in evolutionären Silizium-Schaltungen gleichsam selbst programmiert: “ein physisches Gerät, das in einer Umgebung aus lauter physischen Geräten arbeitet und nur derselben Beschränkung seiner Ressourcen wie sie untersteht. Software im üblichen Sinn [...] gäbe es nicht mehr.”[16]

Ich interpretiere, in einer archäologischen Fussnote, diese hypothetische Vision als einen Affekt. Visionen sind Affekte, zumal wenn sie, wie diese von Friedrich Kittler, auf jede quantenmechanische Betrachtung der Beschreibung von Silizium-Chips verzichtet und damit die Materialität der Technologie links liegen lässt, welche seit den 1960er Jahren die digitale Welt hervorbringt. Es ist auch ein Affekt der Invisibilisierung, der Unsichtbarmachung der Unfertigkeit des Computers selbst, ohne aber dessen Unfertigkeit zu leugnen, sondern im

---

[15] in: Kittler, Friedrich (1993): Es gibt keine Software, in: ders.: Draculas Vermächtnis. Technische Schriften. 225-242, Leipzig, 237.

[16] ebd, 241.

Gegenteil, indem durch ihre Skandalisierung und visionäre Eskamotierung. Bernhard Siegert, der Kittler-Schüler von Freiburg bis Berlin, hat jüngst diese Vision ex post für eine “antihumanistische [...] Medientheorie” erklärt, wogegen ich den Einwand erheben möchte, dass Maschinen, die sich selbst programmieren, immer schon auch zutiefst romantische Visionen waren, wie sie bereits im Gefolge von Lichtenberg, Novalis, Hoffmann bis hin zu Hoffmansthal gedacht wurden, selbstaufschreibende Chiffren des Natur. Insofern sind sie, aus meiner Sicht, alles andere als anti-humanistisch oder jedenfalls nicht anti-humanistischer als Schelling’s Weltseele. Tatsache bleibt überdies, dass man bis heute solche sogenannt “transfiniten” Turing-Maschinen nicht bauen kann, bei denen die Hardware generisch sich selbst programmiert. Insofern bleibt Friedrich Kittler dieser antimetaphysische Metaphysiker, der er wohl immer war, was die Berechtigung seiner scharfen Kritik an der Software-Industrie als neuer gouvernementaler Regierungsmacht im Sinne Foucaults nicht im mindesten schmälert. Sie ist heute noch viel aktueller als sie es je war.

Denn heute sehen wir besser, was den Affekt-Effekt des Computers über die Jahrzehnte so erfolgreich gemacht hat. Ich will hier zum Schluss nur andeuten, welche zwei Stränge einer Genealogie dieses Effekts zu sehen sind. Beide haben ihre Wurzeln in Utah, im Kontext des ersten graphischen Zeichenprogramms “Sketchpad”, das Ivan Sutherland dort im Rahmen seiner Dissertation Anfang der 1960er Jahre entwickelt hatte, ein bereits objekt-orientiertes Graphik-Programm, das einen virtuell unbegrenzten Bildschirm mit geometrischen Objekten bespielen konnte. Fast mehr noch als “Sketchpad” wurde ein theoretisches Paper Ivan Sutherlands einflussreich, das den Titel “The Ultimate Display” trägt und fordert, den Computer, weg von allen Screens und Bildschirmen, in den Raum verlagert werden müsse, “innerhalb dessen der Computer die Existenz der Materie kontrollieren kann. Ein Stuhl, der in einem solchen Raum ausgestellt ist, wäre gut genug, um darin zu sitzen. Handschellen, die in einem solchen Raum ausgestellt sind, könnten beängstigend wirken, und eine Gewehrkugel, die einen solchen Raum durchzischt, wäre fatal.”[17] Sutherland entwirft hier nichts anderes als das heutige ubiquitous computing samt aller Daten-Prothesen und -Sensoren, wearable computings, Datenbrillen, Tracking Armbänder, Data Gloves und dergleichen mehr. Sein Papier triggerte zum Beispiels Minsky’s Forschungen im MIT wesentlich mit an.

---

[17] in: Sutherland, Ivan E. (1965): The Ultimate Display, Proceedings of IFIP Congress, pp. 506-508, 507.

Der zweite genealogische Strang ginge von der Gruppe um Alan Kay aus, dessen Doktorvater 1968 der kaum ältere Ivan Sutherland war. Kays Team und die Kinder der “Learning Reseach” Group entwickelten im Xerox-PARC Projekt der 1970er Jahre jenes “Graphical User Interface”, das heute alle unsere PC’s beherrscht, Icons, Drop-Down-Menüs, Überlappende Fenster. Zunächst wurde diese GUI im MacIntosh 1984 realisiert, dann Ende in der 1980er mit Windows bei Microsoft. Bei dieser Affektion einer “User Illusion”, die Kay erzielen wollte, ging es um die Ausbeutung kindlicher Affekte gemäß der Piaget’schen Entwicklungstheorie, in ihrer vereinfachten Version von Jerome Bruner, was zu der Formel führte “Doing with Images makes symbols” und im Ergebnis jene Kinderklötzchen auf den Bildschirm zauberte, die wir heute Icons nennen, auf die wir intuitiv doppelt klicken, worauf sich Bilder von Seiten öffnen, auf die wir dann Vorträge schreiben können, was dann unsere erste und für heute auch letzte symbolische Handlung wäre.

Nur ein Satz zum Schluss: Beide Entwicklungsstränge der affektiven Computereffekte konvergieren seit gut einem Jahrzehnt im Smartphone. Es implementiert eine gouvernementale User Affektion über das grafische Interface, und implementiert zugleich die Haptik des ubiquitous computing. In dieser Ökosensorik entfaltet sich die neoliberale Doppelnatur des Smartphone-Gebrauchs, nämlich die Verbindung von Ich-Unternehmerschaft und Allmachts-Konsumismus. Hier ist der Computer buchstäblich verschwunden, nämlich in der “Sandbox”. “Sandkasten”, so heisst die Applikation, die wir weder aufrufen noch schließen können. Sie verschließt uns den dahinter liegenden Computer versperrt, der das Smartphone simuliert.

Digitale User-Illusionen und glasglatte Streichel-Haptiken erzeugen Affekte, die es zuvor so nicht gab. Aber das macht sie noch nicht zu primordialen Phänomenen. Es sind keine metaphysischen Superjekt-Entitäten im Sinne von Whiteheads Prozess-Metaphysik. Im Gegenteil, die Effekte des Verschwinden-Lassens des Computers in der digitalen Kultur sind und bleiben Effekte der Verbergung von Imperfektion, die eine gouvernementale Macht erzeugen und uns umso mehr Anlass geben, ihre Regime offen zu legen.