

Publiziert in: Arns, Inke / Goller, Mirjam / Strätling, Susanne / Witte, Georg (Hg.): Kinetographien.
Bielefeld: Aisthesis 2004, S. 57-78

Inke Arns (Berlin)

Texte, die (sich) bewegen: zur Performativität von Programmiercodes in Netzkunst und Software Art

„Der Mensch fängt an zu begreifen, wie die
komplizierteste aller seiner Maschinen
auseinanderzunehmen und wieder
zusammensetzen ist: Die Sprache.“¹
(Italo Calvino)

„Software is mind control.
Come and get some.“
(Slogan der Londoner Künstlergruppe
I/O/D für ihr Programm *Web Stalker*²)

Code, oder genauer: Programmiercode bewirkt, so Friedrich Kittler, „dass aus etwas etwas anderes wird, nicht nur wie bei Metaphern etwas anders heißt.“³ Insofern erscheint Programmcode als „codierte Performativität“⁴ oder *(auto-)generativer Text* geradezu prädestiniert für eine eingehendere Untersuchung im Kontext unserer Auseinandersetzung mit *Kinetographien*, Bewegungs-Schriften. Es ist gerade der *generative, mobilisierende* Aspekt des Computercodes, der diesen Code von literarischer oder alltäglicher Sprache unterscheidet:

Codes sollten daher einzig Alphabete im Wortsinn der modernen Mathematik heißen, eindeutige und abzählbare, ja, möglichst kurze Folgen von Symbolen also, die dank einer Grammatik mit der unerhörten Fähigkeit begabt sind, sich gleichwohl selbst unendlich zu vermehren: Semi-Thue-Gruppen, Markowketten, Backus-Naur-Formen usw. Das und nur das unterscheidet solche modernen Alphabete vom vertrauten, das unsere Sprachen ja zwar auseinanderlegte und Homers Gesänge schenkte, aber keine Technikwelt zum Laufen bringt wie heutzutage Computercode.⁵

¹ Calvino, Italo, zit. in Grassmuck, Volker: *Vom Animismus zur Animation. Anmerkungen zur künstlichen Intelligenz*. Hamburg 1988, 86.

² I/O/D, *Webstalker* (1997), <http://www.backspace.org/iod> [19.11.2003].

³ Kittler, Friedrich: Code oder wie sich etwas anders schreiben lässt. In: Schöpf, Christine / Stocker, Gerfried (Hg.): *Ars Electronica 2003: Code - The Language of Our Time*, Ostfildern 2003, 15-19, hier 18, http://www.aec.at/de/archiv_files/20031/FE_2003_Kittler_de.pdf [17.11.2003].

⁴ Grether, Reinhold: The Performing Arts in a New Era. In: *Rohrpost*, 26.7.2001, <http://buug.de/pipermail/rohrpost/2001-July/000353.html> [19.11.2003].

⁵ Kittler, Code, 18.

Auch Florian Cramer, Ulrike Gabriel und John F. Simon Jr. interessieren sich insbesondere für die Algorithmen – „de[n] eigentliche[n] Code, der erzeugt, was man sehen, hören und spüren wird“⁶. Für sie ist der „vielleicht faszinierendste Aspekt der Computertechnik“, dass Code – ob als Textfile oder Binärzahl – maschinell ausführbar wird: „Ein harmloses Stück Text kann das System stören, verändern oder gar abstürzen lassen.“⁷ Im Code – und nicht in den von ihm lediglich generierten „fluiden“ Oberflächen – findet sich somit eine paradoxe Verschränkung von Statik und Dynamik, in Form einer mobilisierenden Statik, einer dynamisierenden, in Bewegung setzenden Notation, einer rekursiven Linearität. Programmcode als „codierte Performativität“ hat zudem unmittelbare, auch politische Konsequenzen auf die virtuellen Räume, in denen wir uns zunehmend bewegen: Hier ist der Code Gesetz, „Code is Law“⁸. Kurz gefasst: Die Frage nach der Performativität des Code als dem eigentlichen Text, der bewegt, erscheint interessanter als die Frage nach den von ihm affizierten Oberflächen, weil ambivalenter, und vielleicht auch bewegender.

Eine solche Auseinandersetzung mit dem Programmiercode und seiner ubiquitären – jedoch unsichtbaren – Präsenz findet sich in der sogenannten „Softwarekunst“. Dieser Begriff wurde 2001 erstmalig von dem Berliner Medienkunstfestival *transmediale* definiert und als Wettbewerbskategorie eingeführt.⁹ Softwarekunst, die von anderen AutorInnen auch als „experimentelle“¹⁰ oder „spekulative Software“¹¹ sowie als „nicht-pragmatische“ und „nicht-rationale“¹² Software bezeichnet wird, umfasst nach der von der *transmediale*-Jury formulierten Definition Projekte, deren wesentliches künstlerisches Material Programmcode ist, oder die sich mit dem kulturellen Verständnis von Software auseinandersetzen. Software-

⁶ Jury Statement “Software”, Transmediale 2001, <http://www.transmediale.de/01/de/software.htm> [19.11.2003].

⁷ Jury Statement “Software”, Transmediale 2001, <http://www.transmediale.de/01/de/software.htm> [19.11.2003].

⁸ Lessig, Lawrence: *Code and other Laws of Cyberspace*. New York 1999, <http://www.code-is-law.org/> [19.11.2003].

⁹ Für ein frühes, programmatisches Konzeptpapier über Softwareprogrammierung und Kunst vgl. Cox, Geoff / McLean, Alex / Ward, Adrian: *The Aesthetics of Generative Code* (2000), <http://generative.net/papers/aesthetics/> [18.11.2003]. Eine ausführliche Vorstellung künstlerischer Software-Programme, Viren, Absturzcodes, Spiel-Modifikationen, u.a. findet sich bei Baumgärtel, Tilman: Experimentelle Software. Zu einigen neueren Computerprogrammen von Künstlern. In: *Rohrpost*, 21.5.2000, <http://amsterdam.nettime.org/Lists-Archives/rohrpost-0005/msg00138.html> [18.11.2003]. Der Versuch einer formalen Definition und historischen Archäologie von Softwarekunst anhand literarischer und künstlerischer Beispiele findet sich bei Cramer, Florian: Concepts. Notations. Software. Art, 23.3.2002, http://userpage.fu-berlin.de/~cantsin/homepage/writings/software_art/concept_notations/concepts_notations_software_art.html [19.11.2003].

¹⁰ Baumgärtel, Experimentelle Software.

¹¹ Matthew Fuller z.B. unterscheidet „kritische“, „soziale“ und „spekulative Software“. Vgl. Fuller, Matthew: Behind the Blip: Software as Culture. In: *Nettime*, 7.1.2002, <http://amsterdam.nettime.org/Lists-Archives/nettime-l-0201/msg00025.html> [19.11.2003].

¹² Olga Goriunova und Alexei Shulgin definieren “künstlerische Software” als “nicht-pragmatisch” und nicht-rationale”: “[I]f conventional programs are instruments serving purely pragmatic purposes, the result of the work of artistic programs often finds itself outside of the pragmatic and the rational.” (Goriunova, Olga / Shulgin, Alexei: *Artistic Software for Dummies and, by the way, Thoughts About the New World Order*. In: *Nettime*, 26.5.2002, <http://amsterdam.nettime.org/Lists-Archives/nettime-l-0205/msg00169.html> [19.11.2003]).

Code wird hier nicht als pragmatisch-funktionales Werkzeug zur Bedienung der ‚eigentlichen‘ künstlerischen Arbeit verstanden, sondern als generatives Gestaltungsmaterial maschineller und sozialer Prozesse. Software-Kunst kann dabei das Resultat einer autonomen und formalen kreativen Praxis sein, sie kann sich aber auch kritisch und collagierend auf existierende Software und die technologische, kulturelle oder soziale Bedeutung von Software beziehen:¹³ „Code can be diaries, poetic, obscure, ironic or disruptive, defunct or impossible, it can simulate and disguise, it has rhetoric and style, it can be an attitude“¹⁴, so die emphatische Definition von Florian Cramer und Ulrike Gabriel, beide Mitglieder der *transmediale*-Jury 2001.

Eine weitere, von Alan Sondheim im selben Jahr (allerdings im Kontext experimenteller Poesie) geprägte Bezeichnung für die künstlerische bzw. künstlerisch-literarische Auseinandersetzung mit Programmiercodes sind die sogenannten „Codeworks“¹⁵ der Netzkunst und -literatur.¹⁶ Diese benutzen formalen ASCII-Instruktionscode bzw. dessen Ästhetik, ohne jedoch auf die von ihm geschaffenen Oberflächen und multimedialen graphischen Benutzerinterfaces zu rekurrieren. Die Arbeiten von Jodi, Netochka Nezvanova und mez, die in diesem Kontext vorgestellt werden sollen, rufen die Existenz eines normalerweise durch die graphischen Oberflächen verdeckten, unsichtbaren „postoptischen Unbewussten“ ins Gedächtnis, analog der mechanischen Aufnahmetechniken, die es laut Walter Benjamin ermöglichten, das vom Bewusstsein verdrängte „optische Unbewusste“ aufzuzeichnen.

Fluide Oberflächen = mobile Texte?

Angesichts der heute möglich gewordenen „Mobilisierung“ von Texten und Bildern – durch die technische Implementierung von Beweglichkeit, Fluidität und Dynamik – ist oft von einem „Verlust der Einschreibung“ die Rede. Hypertextuelle und -mediale Texte bzw. Werke zeichnen sich, so die verbreitete Behauptung, zunehmend durch permanente Veränderung und

¹³ Vgl. http://www.transmediale.de/04/pdf/tm04clubtm04_formular_ausschreibung.pdf [17.11.2003]. Zum Thema Software Art vgl. aktuell auch die Podiumsdiskussion während der *transmediale.03* (Künstlerhaus Bethanien, 4.2.2003), <http://www.softwareart.net/> [19.11.2003], sowie Goriunova, Olga / Shulgin, Alexei (Hg.): *Read_Me 2.3 Reader - about software art*. Helsinki 2003, http://www.m-cult.org/read_me [17.11.2003].

¹⁴ Cramer, Florian / Gabriel, Ulrike: Software Art. In: Broeckmann, Andreas / Jaschko, Susanne (Hg.): *DIY Media - Kunst und digitale Medien: Software - Partizipation - Distribution. Transmediale.01*. Berlin 2001, 29-33, hier: 33.

¹⁵ Sondheim, Alan: Codework. In: *American Book Review*, Vol. 22, Issue 6 (September/October 2001), <http://www.litline.org/ABR/PDF/Volume22/sondheim.pdf> [19.11.2003].

¹⁶ Ausführlich zur Netzkunst vgl. Arns, Inke: *Netzkulturen*. Hamburg 2002; Baumgärtel, Tilman: *[net.art]. Materialien zur Netzkunst*. Nürnberg 1999; zur Netzliteratur vgl. Heibach, Christiane: *Literatur im Internet: Theorie und Praxis einer kooperativen Ästhetik*. Phil.-Diss., Universität Heidelberg, 2000, http://www.netzliteratur.net/heibach/heibach_diss.pdf [19.11.2003].

Flüchtigkeit und nicht mehr durch materiell festgeschriebene Statik aus. Endlich, so frohlocken MedientheoretikerInnen und -philosophInnen, sei man mit genuin dynamischen Texten konfrontiert und müsse sich nicht mehr mit stillgestellten, arretierten und immobilisierten Texten zufrieden geben.

So ergibt sich z.B. für Martin Burckhardt hinsichtlich der elektromagnetischen Schrift eine „Verflüssigung des tradierten Schriftbegriffs“¹⁷. Wenn Sybille Krämer von der „Fluidität des Wortes“ spricht und dieses Wort als Medium bezeichnet, „dessen Daseinsweise Flüchtigkeit und Fluktuanz ist durch und durch“¹⁸, ließe sich – spinnt man das Flüchtigkeitsparadigma weiter – vermuten, dass Netzkunst bzw. –literatur die Eigenschaften des gesprochenen Wortes annehme. Auch Umberto Eco konstatiert, dass „the infinity or at least the indefinite abundance of interpretation is due not only to the initiative of the reader, but rather to the *physical mobility of the text itself*, a text that is produced just in order to be re-written.“¹⁹ Und Peter Matussek spricht hinsichtlich der „Oberflächeneffekte der Software“ – also der dynamischen Datenpräsentationen durch Inszenierung von Information und Animation – sogar von einem „’performative turn’ graphischer Benutzerschnittstellen“²⁰.

Es scheint sich also hinsichtlich nun auch technisch realisierbarer Nonlinearität, Interaktivität und hypertextueller Interkonnektivität folgende Tendenz für ein neues Textverständnis durchzusetzen: Alles fließt und dies ist nur möglich durch neue Medien. Diese Sichtweise verschweigt jedoch, dass Texte nicht erst mit der technischen Implementierung von Bewegung mobil werden. Darauf hat spätestens die Intertextualitätstheorie mit ihrem Verweis auf den Verbindungs- bzw. Netzcharakter von Texten in den 1960er Jahren hingewiesen. Auch sind sich selbst schreibende bzw. generierende Texte weder in der Literatur, noch in der Kunst unbekannt.²¹ „Ich drückte die linke Hand auf das Titelblatt und schlug das Buch auf, den Daumen fest an den Zeigefinger gepresst. Alles war zwecklos: Immer schoben sich einige Blätter zwischen Titelblatt und Hand. Es war, als brächte das Buch sie hervor“, heißt es in Jorge Luis Borges Erzählung über „El libro de arena“, ein gespenstisch unerschöpfliches, sich bei jedem Aufschlagen veränderndes „Buch aus Sand“. Borges fand mit dieser Parabel ein literarisches Bild für die

¹⁷ Burckhardt, Martin: Unter Strom. Der Autor und die elektromagnetische Schrift. In: Sybille Krämer (Hg.): *Medien Computer Realität*. Frankfurt/Main 2000, 27-54, hier: 27.

¹⁸ Krämer, Sybille: Sprache-Stimme-Schrift. in: *Paragrana* 7 (1997), 1, S. 45.

¹⁹ Eco, Umberto: Books, Text and Hypertext. A Talk by Umberto Eco. In: *Rampike* 10/2 (1999), meine Hervorhebungen.

²⁰ Matussek, Peter: Performing Memory. Kriterien für einen Vergleich analoger und digitaler Gedächtnistheater. In: *Paragrana* 10 (2001), H. 1, S. 291-320. [Teil A], http://141.20.150.19/PM/Pub/A_40.pdf [19.11.2003].

²¹ Vgl. dazu Florian Cramers Forschungen zu literarischen Permutationen des Barock, die zum Teil auf seiner Website dokumentiert sind (<http://userpage.fu-berlin.de/~cantsin/homepage/> [19.11.2003]), sowie seinen

Unmöglichkeit, dieselbe Seite zweimal zu lesen: Man „blättert, stößt auf Neues, will es festhalten, sich vergewissernd zurückschauen, aber die vorhergehenden Seiten vervielfachen sich bis ins Unendliche eines Labyrinths möglicher Einblicke und lösen sich schließlich auf wie feiner Sand, der zwischen den Fingern zerrinnt. Aus zeitlicher Distanz ist nichts, wie es einmal war.“²² Hier deutet sich bereits die prä-technische Mobilität von Texten an.

Was uns jedoch interessieren soll, ist nicht die Problematisierung eines statischen oder dynamischen Textverständnisses, sondern – ganz im Sinne unseres *Kinetographien*-Konzeptes – die präzisere Herausarbeitung einer Konfiguration bewegender und bewegter Schriften in Netzkunst- und literatur. Dazu soll zwischen multimedialen, dynamischen Oberflächen und Texten bzw. algorithmischen Steuercodes, die diese Oberflächen generieren, unterschieden werden. Ein Verbleiben an der Oberfläche würde die Tatsache verkennen, dass der Computer kein *Bildmedium*, darauf hat Florian Cramer hingewiesen, sondern essentiell ein *Schriftmedium* ist, an das alle möglichen audiovisuellen Ausgabemedien anschließbar sind.²³

Programmiercodes als performative Texte

Die oft formulierte These vom „Verlust der Einschreibung“ mit ihrem ausschließlichen Fokus auf den Oberflächentext als *dem* „Text“ von Netzkunst bzw. -literatur geht, so meine These, wenn nicht gar von einer falschen, so doch von einer *nicht genügend präzisen* Fragestellung aus. Es reicht meiner Ansicht nach nicht aus, hinsichtlich der „Oberflächeneffekte der Software“ – also der dynamischen Datenpräsentationen durch Inszenierung von Information und Animation – von einem „’performative turn’ graphischer Benutzerschnittstellen“²⁴ zu sprechen, denn diese Sichtweise bleibt zu sehr einer unterstellten Performativität eben jener Oberflächen verhaftet. Vielmehr sollte man bei der Betrachtung von Software- und Netzkunstprojekten (wie auch bei Software allgemein) von mindestens zwei Texten ausgehen: einem „Phäno“- und einem „Genotext“. Die Oberflächeneffekte des Phänotextes, z.B. sich bewegende Texte, werden durch andere, unter den Oberflächen liegende „effektive“ Texte,

Vortrag „Wechselsatz und Wechselrad in Quirinus Kuhlmanns XLI. *Libeskuß*“ auf der *Kinetographien*-Konferenz (2001).

²² Haupt, Gerhard: Resistance-images. Selbst(er)findung als Selbstbehauptung. In: *Pat Binder: Zapping*. Institut für Auslandsbeziehungen, Berlin 1996, 8-13, <http://pat-binder.de/de/txt/haupt-ifa-p.html> [18.11.2003].

²³ „Es gibt im Computer nichts als Schrift, woraus folgt, daß Schrift, Text der Schlüssel zum strukturellen Verständnis des Computers und der Digitalisierung analoger Zeichen ist.“ (Cramer, Florian: Für eine Textwissenschaft des Digitalen. Typoskript, Vortrag auf dem Germanistentag Erlangen, 1.10.2001, 2, http://userpage.fu-berlin.de/~cantsin/homepage/writings/theory/germanistentag_2001//textwissenschaft_des_digitalen.pdf [19.11.2003]).

den Programmcodes oder Quelltexten, hervorgerufen und gesteuert. Bei diesen (normalerweise) unsichtbaren Programmiercodes handelt es sich somit um illokutionäre Sprechakte, insofern, als in ihnen ‚Sagen‘ und ‚Tun‘ zusammenfallen, diese „handlungsmächtigen“ Sprechakte also keine Beschreibung oder Repräsentation von etwas sind, sondern direkt affizieren, in Bewegung setzen, Effekte zeitigen.

Friedrich Kittler verwies diesbezüglich bereits in seinem Text „Die Schrift des Computers: A license to kill“ auf den doppeldeutigen Begriff der „Kommandozeile“, ein Zwitterwesen, das heute in den meisten Betriebssystemen durch graphische Benutzeroberflächen fast vollkommen verdrängt worden ist. Noch vor 20 Jahren waren jedoch alle Benutzerschnittstellen und Editoren kommandozeilenorientiert, bzw. man konnte zwischen verschiedenen Modi hin- und herwechseln. Während im Textmodus die Return-Taste zu einem Zeilenwechsel führt, verwandeln sich eingegebene Texte plus Return-Taste im Kommandozeilenmodus in potentielle Befehle: „[U]nd was geschrieben stand fand tatsächlich statt.“²⁵ – Friedrich Kittler beschreibt diesen Unterschied folgendermaßen:

Im Computer [...] fallen, sehr anders als in Goethes *Faust*, Wort und Tat zusammen. Der säuberliche Unterschied, den die Sprechakttheorie zwischen Erwähnung und Gebrauch, zwischen Wörtern mit und ohne Anführungszeichen gemacht hat, ist keiner mehr. kill im Kontext literarischer Texte sagt nur, was das Wort besagt, kill im Kontext der Kommandozeile dagegen tut, was das Wort besagt, laufenden Programmen oder gar dem System selbst an.²⁶

Diese Wirkmächtigkeit oder Performativität der Computerschrift, die sich auch im Bedeutsamwerden kleinster Buchstabenpartikel äußert (Burckhardt und Kittler verweisen auf die orthographische Strenge von Programmiersprachen²⁷), soll hier nun über die Kommandozeile hinaus in ihrer Form als maschinenlesbarer Code, menschenlesbare Programmiersprache bzw. –schriften²⁸ – und in ihrem Bezug zur graphischen Oberfläche untersucht werden.

²⁴ Matussek: *Performing Memory*, 291-320.

²⁵ Kittler, Friedrich: *Die Schrift des Computers. A License to Kill*, 2 [undatiertes Typoskript].

²⁶ Kittler: *Die Schrift des Computers*, 4.

²⁷ Vgl. weiter unten.

²⁸ Die Oppositionen Sprache/Schrift, Mündlichkeit/Schriftlichkeit und Programmiersprache/Code finden hier keine weitere Beachtung, denn sie sind für die vorliegende Argumentation nicht von Bedeutung. Höhere *Programmiersprachen* (d.h. der Quellcode) sind menschenles- und schreibbar. Diese müssen jedoch, um für die Maschine lesbar zu werden, von einem Compiler in ausführbaren binären *Code* übersetzt werden. Der Unterschied zwischen textuellem und numerisch-binärem Code kann hier aber vernachlässigt werden, gerade weil das eine nur eine maschinenlesbare Interpretation bzw. Version des anderen ist. Die Sprechakttheorie bezieht sich explizit auf *gesprochene* Sprache, während Programmiersprachen, die eigentlich *Programmierschriften* heißen müssten, schriftlich verfasst sind. Trotz (oder gerade wegen) ihrer Schriftlichkeit

In einer Reihe von Vorlesungen, die John Langshaw Austin (1911-1960) 1955 an der Harvard University unter dem Titel *How To Do Things With Words*²⁹ (dt.: *Zur Theorie der Sprechakte*) hielt, führte er den bahnbrechenden Gedanken aus, dass sprachliche Äußerungen keineswegs nur dem Zweck dienen, einen Sachverhalt zu beschreiben oder eine Tatsache zu behaupten, sondern dass mit ihnen stets Handlungen vollzogen werden. „Was Sprecher von Sprachen intuitiv immer schon gewußt und praktiziert haben,“ so schreibt Erika Fischer-Lichte, „wurde hier von der Sprachphilosophie zum ersten Mal formuliert: dass Sprache nicht nur eine referentielle Funktion erfüllt, sondern immer auch eine performative.“³⁰ Austins Sprechakttheorie begreift Sprechen also grundsätzlich als Handeln und beobachtet dabei ein Sprechen, das nicht erst durch seine Wirkung effektiv ist, sondern bereits durch sich selbst. Genau hier trifft sich die Sprechakttheorie mit der unterstellten Performativität des Code: „[W]enn ein Wort nicht nur etwas benennt, sondern etwas performativ herbeiführt und zwar genau das, was es benennt“³¹.

Programmiercodes als geglückte (effektive) Illokutionen

Austin unterscheidet nun in allen Sprechakten drei verschiedene linguistische Akte. Den *lokutionären Akt* bestimmt er als den propositionalen Gehalt, der wahr oder falsch sein kann. Er soll uns in diesem Zusammenhang nicht weiter interessieren. *Illokutionäre Akte* sind Handlungen, die kraft der Worte ausgeführt werden. Sie sind dadurch bestimmt, dass jemand, indem er etwas sagt, gleichzeitig etwas tut (die Aussage des Richters „ich verurteile Sie“ ist keine Absichtserklärung, sondern ein Tun). Bezeichnung und Ausführung fallen zusammen, indem die Bezeichnung „geäußert wird, führt sie selbst eine Tat aus“³². Illokutionäre Sprechakte rufen also Effekte hervor und können gelingen bzw. misslingen, je nachdem ob bestimmte extralinguistische Konventionen erfüllt werden. *Perlokutionäre Akte* sind dagegen solche Äußerungen, die eine Kette von Folgen auslösen. Das Sagen und die hervorgerufenen Wirkungen fallen zeitlich nicht zusammen. Wie Judith Butler bemerkt, sind die „Folgen nicht dasselbe wie der Sprechakt, sondern eher die Ergebnisse oder das ‚Nachspiel‘ der Äußerung.“³³ Sie bringt diesen Unterschied auf folgende prägnante Formel: „Während

handelt es sich jedoch, so die These, um „wirkungs-“, oder „handlungsmächtige“ Texte, die keine Beschreibung oder Repräsentation von etwas sind, sondern direkt affizieren, in Bewegung setzen, Effekte zeitigen.

²⁹ Austin, John Langshaw: *How to Do Things with Words*, Claredon Press 1962 (dt.: *Zur Theorie der Sprechakte*, Stuttgart 1979).

³⁰ Fischer-Lichte, Erika: Auf dem Weg zu einer performativen Kultur. In: *Paragrana* 7 (1998) 1, 13-29.

³¹ Butler, Judith: *Hass spricht. Zur Politik des Performativen*. Berlin 1998, 67.

³² Butler: *Hass spricht*, 67.

³³ Butler: *Hass spricht*, 31.

illokutionäre Akte sich mittels Konventionen vollziehen, vollziehen sich perlokutionäre Akte mittels Konsequenzen. Diese Unterscheidung beinhaltet also, dass illokutionäre Sprechakte ohne zeitlichen Aufschub Effekte hervorrufen, dass hier das ‚Sagen‘ dasselbe ist wie das ‚Tun‘ und dass beide gleichzeitig erfolgen.³⁴ Insofern, als hier ‚Sagen‘ und ‚Tun‘ zusammenfallen, ließen sich also Programmiercodes als illokutionäre Sprechakte bezeichnen.

Sprechakte können nach Austin auch Handlungen sein, ohne jedoch unbedingt effektiv sein zu müssen (d.h. „glücken“ zu müssen). Scheitern oder missglücken diese Handlungen, stellen sie verfehlte performative Äußerungen dar. Ein Sprechakt ist, auch wenn er sprachliches Handeln ist, also nicht immer ein *effektiver* Akt. „Eine geglückte performative Äußerung ist [jedoch] dadurch definiert,“ so Judith Butler, „dass ich die Handlung nicht nur ausführe, sondern damit eine bestimmte Kette von Effekten auslöse.“³⁵ Programmiercodes machen, ganz pragmatisch betrachtet, nur als geglückte performative Äußerungen Sinn; lösen sie keine Effekte aus (egal, ob diese erwünscht oder unerwünscht sind), sind sie nicht ausführbar, sind sie schlicht und ergreifend überflüssig. Code macht im Kontext von funktional-pragmatischer Software nur als ausführbarer Code Sinn.³⁶

Es ist also der Code, der performativ ist, und nicht die (arbiträren, medialen) Oberflächen, die er erzeugt und steuert. Im Code fallen ‚Sagen‘ und ‚Tun‘ zusammen, insofern diese handlungsmächtigen Sprechakte keine Beschreibung oder Repräsentation von etwas sind, sondern diese direkt affizieren, in Bewegung setzen, Effekte zeitigen. Diese Macht oder Wirkmächtigkeit des Code bzw. seine Performativität zeigt sich auch in der orthographischen Strenge von Programmiersprachen: „Während ein Text auch noch in verstümmelter Form Sinn macht, bricht die Übertragung ab, sobald nur ein einziges Glied aus der Kette herausfällt. Genau das ist es ja, was unsere heutigen Computerprogramme ausmacht. Wenn in einem solchen Programm eine einzige Zeile fehlt, wenn ein einziger Buchstabe falsch geschrieben ist, so ist der Regelkreislauf unterbrochen – und das Programm kann nicht ausgeführt werden.“³⁷ Friedrich Kittler konstatiert diesbezüglich: „Programmiersprachen [haben] all das, was an Strenge verloren gegangen ist, übernommen [...]. Unterscheidungen wie die zwischen Komma und Semikolon, auf dem Papier fast verschwundene Nuancen also, kehren in despotischer Härte wieder. [H]eute kann ein fehlendes oder mit dem Komma vertauschtes

³⁴ Butler: *Hass spricht*, 31f.

³⁵ Butler: *Hass spricht*, 31.

³⁶ Innerhalb der Softwarekunst macht jedoch durchaus auch nicht-ausführbarer Code Sinn; dazu weiter unten.

³⁷ Burckhardt: *Unter Strom*, 38.

Semikolon ganze Programme oder gar Betriebssysteme zum Absturz bringen.“³⁸ Zu Recht wünscht sich Friedrich Kittler daher schon seit geraumer Zeit einen neuen „Computeralphabetismus“, der sich im Sinne einer kritischen Medienkompetenz der Codes annimmt. Und Florian Cramer fordert diesbezüglich die Literaturwissenschaft auf, sich wieder auf ihre Fähigkeiten als Textwissenschaft zu konzentrieren.

Sybille Krämer definiert Performativität als *Verkörperung* im Gegensatz zur Darstellung, als *Präsentation* im Gegensatz zur Repräsentation und als *Epiphanie* bzw. *Gegenwärtigkeit* anstelle einer Stellvertreterschaft oder Vergegenwärtigung – eine Definition, die hier vielleicht nutzbar zu machen wäre. *Verkörperung* und *Gegenwärtigkeit* sind sicherlich auch der Grund für die Gewalt des Eigen- bzw. Markenamens im Internet, wie Birgit Richard in ihrer Untersuchung der strukturellen, bilderlosen Gewalt im Netz zeigt. „Man könnte fast glauben,“ so Richard, „man trete in ein magisches Zeitalter ein, in dem die Nennung eines Namens direkte Auswirkungen auf die Realität hat. Der Eigenname gilt hier wie bei einer Verfassung als Gründungstext.“³⁹ Auch Derrida bezeichnet einen solchen institutionellen Gründungsakt als Sprechakt, denn dieser „Akt besteht nicht in einem deskriptiven oder konstatierenden Diskurs, er vollzieht, vollbringt und tut, was er zu tun sagt [...]“.⁴⁰ Der Domainname im Web, der eine Vereinfachung der numerischen IP-Adresse ist, ist mehr als nur ein Name: er hat performative Eigenschaften und er ist ausschließlicher Ort: Es gibt keinen anderen neben ihm.

Performativität läuft so letztlich auf die magische Ineinssetzung von Zeichen und Bezeichnetem hinaus, die sich auch in der Entwicklung der Zahlen in der Mathematik findet. Während George Boole in seinen *Investigations on the Laws of Thought* (1854) Null und Eins nicht mehr als *Repräsentanten* von einem Ding begreift, sondern zu systemischen Markern für An- bzw. Abwesenheit machte⁴¹, erklärte der Mathematiker David Hilbert in „einer radikalen Tautologie [...] die Zeichen, mit denen die Mathematik operiert, aber auch nur operiert hatte, zu ihrer Sache selbst.“⁴² – Mit den *computable numbers* verschwindet die Abbildbeziehung der Zahlen zur Welt; „Sache der Mathematik sind“, so Friedrich Kittler, seit

³⁸ Kittler: Die Schrift des Computers, 8.

³⁹ Richard, Birgit: Am Anfang war das Wort: Domain war's! Zur Gewalt des Eigennamens in virtuellen Welten. In: *Kunstforum International*, Bd. 153, Jan.-März 2001 „Choreographie der Gewalt“, 202-229, hier: 207.

⁴⁰ Derrida, Jacques: Otobiographien. Die Lehre Nietzsches und die Politik de Eigennamens. In: Derrida, Jacques/Kittler, Friedrich: *Nietzsche – Politik de Eigennamens. Wie man abschafft, wovon man spricht*. Berlin 2000, 7-63, hier: 10.

⁴¹ „Booles große Tat besteht darin, dass er die Algebra vom Zahlzeichen löste, dass er die Null und die Eins nicht mehr als *Repräsentanten* von einem Ding begreift, sondern dass er sie zu Markern des *Systems* macht, innerhalb dessen die Dinge erscheinen. Die Null und die Eins sind eigentlich nicht mehr Zahlen, sondern stehen für das System selbst. Für Anwesenheit, Abwesenheit.“ (Burckhardt: Unter Strom, 44).

Hilbert „keine Wesenheiten mehr, die vom Papier lediglich bezeichnet würden; Sache sind gerade umgekehrt die materialen Signifikanten auf dem Papier selber.“⁴³

Bewegung und Stillstand, Dynamik und Statik, Linearität und Non-Linearität gehen im Code ein paradoxales Verhältnis ein: Während der invariable Code als Schrift sich „an einer Geraden [...] entfaltet“⁴⁴ – *scriptum* bedeutet im Lateinischen nicht nur Schrift, sondern auch Linie –, dieser Code somit Linearität und auch eine gewisse Statik verspricht, kann es doch bereits schon auf dieser Ebene nonlineare, dynamische, zyklische Zustände geben – und zwar noch bevor eine Oberfläche ins Spiel kommt, die sich bewegen könnte. Diese Kopplung von Statik und Dynamik im Code ist weniger paradox, wenn man bedenkt, dass es die Performativität des (statischen, linearen) Textes ist, die den Text zum Abarbeiten textinterner Rekursionen und Schleifen anhält. Im Hinblick auf unser *Kinetographien*-Konzept könnte man daher sagen, dass es sich bei Programmcodes nicht um ein Aufzeichnungssystem von Mobilitäten oder Dynamiken handelt, sondern um ein „Mobilisierungs“- bzw. „Immobilisierungssystem“.

Code als Mobilisierungs- bzw. Immobilisierungssystem

Code als „Mobilisierungs“- bzw. „Immobilisierungssystem“ wirkt sich jedoch nicht nur auf graphische Benutzeroberflächen aus. „Codierte Performativität“⁴⁵ hat genauso unmittelbare, auch politische Auswirkungen auf die virtuellen Räume (u.a. des Internet), in denen wir uns zunehmend bewegen: „Programmcode“, so der amerikanische Jurist Lawrence Lessig, „tendiert immer mehr dazu, zum Gesetz zu werden.“⁴⁶ Heute werden Kontrollfunktionen direkt in die Architektur des Netzes, also seinen Code, eingebaut. Diese These stellt Lessig in *Code and other Laws of Cyberspace*⁴⁷ (1999) auf. Am Beispiel des Online-Dienstes AOL macht Lessig eindringlich klar, wie die AOL-Architektur mit Hilfe des sie bestimmenden Codes z.B. jegliche Form von virtueller ‚Zusammenrottung‘ verhindert und eine weitgehende Kontrolle der Nutzer erlaubt. Unterschiedliche Codes erlauben unterschiedliche Grade von Bewegung(sfreiheit): „Die Entscheidung für einen bestimmten Code ist,“ so Lessig, „auch

⁴² Kittler: Die Schrift des Computers, 5.

⁴³ Kittler: Wenn das Bit Fleisch wird. In: Klepper, Martin / Mayer, Ruth / Schneck, Ernst-Peter (Hg.): *Hyperkultur. Zur Fiktion des Computerzeitalters*. Berlin / New York 1996, 150-162, hier: 154. Auch in: Kittler: Die Schrift des Computers, 5.

⁴⁴ Burekhardt: Unter Strom, 35.

⁴⁵ Grether: The Performing Arts in a New Era.

⁴⁶ Lessig, Lawrence, in: futurezone.orf.at: Stalin & Disney - Copyright killt das Internet. In: *Rohrpost*, 30.5.2000, <http://www.nettime.org/Lists-Archives/rohrpost-0005/msg00190.html> [18.11.2003].

⁴⁷ Lessig: *Code and other Laws of Cyberspace*.

eine Entscheidung über die Innovationen, die der Code zu fördern oder zu hemmen imstande ist.“⁴⁸ Insofern mobilisiert bzw. immobilisiert der Code seine Benutzer.

Dieser wirkmächtige Code bleibt jedoch unsichtbar – Graham Harwood bezeichnet diese ‚unsichtbare Welt‘ daher auch als ‚invisible shadow world of process‘⁴⁹. In diesem Sinne könnte man von der Gegenwart als von einem ‚postoptischen‘ Zeitalter sprechen, in dem der Programmcode – den man in Anlehnung an Walter Benjamin auch als ‚Postoptisch-Unbewusstes‘ bezeichnen könnte – zum ‚Gesetz‘ wird. Den Begriff des ‚Postoptischen‘ habe ich in kritischer Auseinandersetzung mit dem Konzept der Ausstellung *ctrl_space* (2000-2001) am ZKM in Karlsruhe entwickelt. Diese Ausstellung, die sich ganz dem Benthamschen *panoptisch-visuellen* Paradigma widmete, stellte die problematische (und hier polemisch formulierte) These auf, dass Überwachung heutzutage nur stattfinden würde, wenn eine Kamera anwesend sei – angesichts der schon heute praktizierten unterschiedlichen Formen von ‚Dataveillance‘ eine unverständliche Auslassung. Der Begriff des *Postoptischen* bezeichnet dagegen all die digitalen Datenströme und (programmierten) Kommunikationsstrukturen und -architekturen, die mindestens ebenso gut zu überwachen sind, aber nur zu einem kleinen Teil aus visuellen Informationen bestehen.⁵⁰

Walter Benjamin definierte das Optisch-Unbewusste in seiner *Kleinen Geschichte der Photographie* als eine unbewusste visuelle Dimension der materiellen Welt, die normalerweise vom gesellschaftlichen Bewusstsein des Menschen herausgefiltert wird und somit unsichtbar bleibt, die aber durch den Einsatz mechanischer Aufnahmetechniken (Photographie und Film: Zeitlupen, Vergrößerungen) sichtbar gemacht werden kann:

Es ist ja eine andere Natur, welche zur Kamera als welche zum Auge spricht; anders vor allem so, dass an die Stelle eines vom Menschen mit Bewusstsein durchwirkten Raums ein unbewusst durchwirkter tritt. Ist es schon üblich, dass einer, beispielsweise, vom Gang der Leute, sei es auch nur im groben, sich Rechenschaft gibt, so weiß er bestimmt nichts mehr von ihrer Haltung im Sekundenbruchteil des ‚Aussehens‘. Die Photographie mit ihren Hilfsmitteln: Zeitlupen, Vergrößerungen erschließt sie ihm. Von diesem Optisch-Unbewussten erfährt er erst durch sie, wie von dem Triebhaft-Unbewussten durch die Psychoanalyse.⁵¹

⁴⁸ Lessig, Lawrence: Die Architektur der Kontrolle: Internet und Macht. In: *Transit/Eurozine*, 28.10.2000, <http://www.eurozine.com/online/partner/austria/transit/issues/2000-01-gs-lessig.html> [18.11.2003].

⁴⁹ Harwood, Graham: Speculative Software. In: Broeckmann, Andreas / Jaschko, Susanne (Hg.): *DIY Media - Kunst und digitale Medien: Software - Partizipation - Distribution. Transmediale.01*. Berlin 2001, 47-49, hier: 47.

⁵⁰ Vgl. dazu auch Druckrey, Timothy: Secreted Agents, Security Leaks, Immune Systems, Spore Wars ... In: *Ctrl_Space. Rhetorics of Surveillance from Betham to Big Brother*, hg. v. Ursula Frohne, Thomas Y. Levin und Peter Weibel, ZKM Karlsruhe, Cambridge, MA 2002, 150-157.

⁵¹ Benjamin, Walter: Kleine Geschichte der Photographie. In: Ders.: *Das Kunstwerk im Zeitalter seiner technischen Reproduzierbarkeit*. Frankfurt/Main 1977, 45-63, hier: 50.

Analog zu Benjamins Definition des Optisch-Unbewussten könnte man heute vielleicht im Computer von der Existenz eines normalerweise durch die grafischen Oberflächen verdeckten „Postoptisch-Unbewussten“ sprechen, das es mittels einer geeigneten Apparatur sichtbar zu machen gilt.⁵² Dieses „Postoptisch-Unbewusste“ wäre als der den Oberflächen zugrundeliegende (für das menschliche Auge meist unsichtbare) Programmiercode zu verstehen, der als codierte Performativität, als algorithmischer Genotext und Tiefenstruktur eben jene für uns sichtbaren Oberflächen generiert, dabei aber selbst unsichtbar bleibt.⁵³

Sichtbarmachen einer unsichtbaren Performativität

Viele künstlerische und netzaktivistische Projekte, die sich seit Ende der 1990er Jahre mit der Politik elektronischer Datenräume (wie z.B. des Internet) auseinandersetzen, setzen daher direkt auf dem *Code* auf und zielen darauf ab, diese technischen Strukturen der *Transparenz* zu entreißen.⁵⁴ Während „Transparenz“ im alltäglichen Verständnis eigentlich für Übersichtlichkeit, Klarheit und für Kontrollierbarkeit durch Einsehbarkeit steht, bedeutet der Begriff in der Informatik das genaue Gegenteil, nämlich Durchsichtigkeit, Unsichtbarkeit und *Information Hiding*. Ist ein System (z.B. eine Oberfläche oder ein Graphical User Interface [GUI]) „transparent“, so bedeutet das, dass der Benutzer es nicht bemerkt, nichts von seiner Funktionsweise mitbekommt. Während dieses *Information Hiding* im Sinne einer

⁵² Einschränkung muss man sagen, dass Benjamins „Optisch-Unbewusstes“ Phänomene unterhalb der menschlichen Wahrnehmungsschwelle bezeichnet, die von dieser zwecks Komplexitätsreduktion ausgeblendet werden müssen. Das heutige „Postoptisch-Unbewusste“ bezeichnet jedoch Programmiercodes, die bewusst verborgen werden, sei es zum Zwecke der Komplexitätsreduktion oder aber zu Zwecken der Schließung und Privatisierung.

⁵³ Vgl. zur Frage des „Unsichtbaren“ auch die Konferenz (*Un-)Sichtbares und Medien* (Hattingen, 23. - 25.10.2002), in deren Vorankündigung es heißt: „Die Tagung handelt vom Umbruch von den Bildmedien (z.B. Fernsehen) zu den Computern. Dieser Umbruch war und ist in seiner Qualität und seinen Folgen umstritten. Die einen betonen, dass der Computer sich grundsätzlich von den Bildmedien unterscheidet, nicht auf Bildern beruhe, sondern auf abstrakten Strukturen wie Texten, Links, Algorithmen und Schrift. Das gelte auch für Datennetze. Das Web z.B. sei ein 'Schriftuniversum', dass die Dokumente im n-dimensionalen Raum anordnet und durch Links vernetzt, nicht mehr aber auch nicht weniger. Die 'Gutenberggalaxis' stünde keineswegs vor ihrem Ende; ganz im Gegenteil, die Explosion der Schrift sprengt die Herrschaft der Bilder. Dagegen sprechen die Apologeten von Multimedia vom Ende der 'Gutenberggalaxis'; Buchkultur und textbezogene Diskussionen verschwinden. Die Schrift verliere an Autorität und Macht. Sie sei ein beschränktes, restriktives System das Multimedia zu überwinden helfe. Hunderttausende kommunizieren und kooperieren jetzt in Datennetzen, versammelt in grafischen, interaktiv und assoziativ zu nutzenden Umgebungen. Bilder gewinnen gegenüber der Schrift an Bedeutung.“ (http://www.hattingen.dgb-bildungswerk.de/vanst2002/ric_prgr.htm, [19.11.2003]).

⁵⁴ KünstlerInnen und NetzaktivistInnen machen mit teils spektakulären Projekten auf die Existenz dieser umkämpften Datensphäre im Internet aufmerksam (z.B. in der *Toywar*-Kampagne), bauen private ECHELON-Systeme (*Makrolab*), entwickeln Werkzeuge zur Verwischung der eigenen Spuren im Internet (*Tracenoizer* von LAN) und thematisieren die zunehmende Einschränkung des öffentlichen Raumes durch Privatisierung von Telekommunikationsinfrastrukturen (Knowbotic Research). Vgl. Arns, Inke: *Netzkulturen im postoptischen Zeitalter*. Vortrag auf der Konferenz *SchnittStellen*, 1. Basler Kongress für Medienwissenschaft, Basel, 20.-23. Juni 2002 [unveröffentlichtes Typoskript], <http://www.mewi.unibas.ch/schnittstellen/index.html> [18.11.2003]; Arns, Inke: *Art Will Be Code, Or It Will Not Be: Medien- und Netzkunst im postoptischen Zeitalter*. Vortrag auf der Konferenz *Save Privacy*, Heinrich-Böll-Stiftung, Berlin, 8.6.2002, <http://www.saveprivacy.org/> [18.11.2003]; Arns: *Netzkulturen*.

Komplexitätsreduktion in vielen Fällen sinnvoll ist, kann es den Benutzer jedoch zugleich in einer falschen Sicherheit wiegen, denn es suggeriert durch seine Unsichtbarkeit eine direkte Sicht auf etwas, eine durch nichts gestörte Transparenz, an die zu glauben natürlich Unsinn wäre: „Far from being a transparent window into the data inside a computer, the interface brings with it strong messages of its own.“⁵⁵ Um diese *message* sichtbar zu machen, gilt es, die Aufmerksamkeit auf die ‚Fensterscheibe‘ selbst zu lenken. So, wie sich durchsichtige Glasfronten von Gebäuden auf Knopfdruck in milchige, halbtransparente Flächen verwandeln, auf die dann per Rückprojektion Bilder projiziert werden können,⁵⁶ so gilt es auch informationstechnische, postoptische Strukturen der *Transparenz* zu entreißen. In den Kommunikationsnetzen ginge es analog dazu darum, die Strukturen ökonomischer, politischer, gesellschaftlicher Machtverteilungen opak werden zu lassen und so sichtbar zu machen. Letztendlich geht es um die Rückführung des informatisch definierten Begriffs der Transparenz (=Durchsichtigkeit des Interface, *Information Hiding*) in seine ursprüngliche Bedeutung von Übersichtlichkeit, Klarheit und Kontrollierbarkeit durch Einsehbarkeit.⁵⁷

Eine zentrale Strategie, derer sich MedienkünstlerInnen bedienen, besteht in der Schaffung und Entwicklung wahrnehmbarer, haptisch-visueller Oberflächen und Interfaces, die normalerweise nicht wahrnehmbare Datenströme und -strukturen sichtbar machen und z.B. in Form statischer oder dynamischer Karto- oder Topographien visualisieren.

Codeworks

Netz- und MedienkünstlerInnen entwickeln jedoch nicht nur eigene künstlerische Interfaces und Oberflächen für die alternative Darstellung von Datenstrukturen, sondern setzen sich auch direkt mit den unter diesen Oberflächen liegenden Programmiercodes auseinander, die performativ Effekte auf den sichtbaren Oberflächen erzeugen. Diese „Textverarbeiter“-Künstler, die sich im Gegensatz zu den „Screendesignern“ mit den strukturellen und technischen Grundlagen des Netzes auseinandersetzen,⁵⁸ arbeiten genau mit dieser Ambivalenz des Code. Die sogenannten „Codeworks“⁵⁹ der Netzkunst (bzw. der Softwarekunst) sind Projekte, die den reinen, „rohen“ formalen ASCII-Instruktionscode bzw. dessen Ästhetik benutzen, ohne jedoch auf die von ihm geschaffenen Oberflächen und

⁵⁵ Manovich, Lev: *The Language of New Media*. Cambridge/Mass. 2001, 65.

⁵⁶ Zu besichtigen z.B. auf der Fassade der ehemaligen VEAG-, heute Vattenfall-Zentrale in der Chausseestr. in Berlin-Mitte.

⁵⁷ Vgl. dazu Arns: Netzkulturen im postoptischen Zeitalter.

⁵⁸ Die Unterscheidung zwischen „Textverarbeitern“ und „Screendesignern“ übernehme ich von Djordjevic, Valentina: Textverarbeiter und Screendesigner: Internet für Netzkünstler leichtgemacht. In: *netz.kunst, Jahrbuch '98-'99*, hg. v. Institut für moderne Kunst Nürnberg 1999, 18-21.

multimedialen graphischen Benutzerinterfaces zu rekurrieren. Die Arbeiten von Jodi, Netochka Nezvanova und mez rufen die Existenz eines normalerweise durch die graphischen Oberflächen verdeckten „Postoptisch-Unbewussten“ ins Gedächtnis.

„M @ z k ! n 3 n . k u n z t . m2cht . fr3!“: „mezangelle“ und „Kroperom“ als künstlerische Appropriationen von Programmiercode

Die australische Netzkünstlerin mez⁶⁰ (Mary-Anne Breeze) und die anonyme Netzentität Netochka Nezvanova (auch bekannt unter den Pseudonymen nn, antiorp und Integer) produzieren seit einiger Zeit neben hypertextuellen Arbeiten bzw. Software zur Echtzeit-Manipulation von Videobildern einfache Textarbeiten, die sie meist in Form von e-mails an Mailinglisten wie Nettime, Spectre, Rhizome, 7-11 oder Syndicate versenden. Sieht man von Attachments und vom zunehmenden Einsatz von html-Text ab, erlaubt das Textmedium e-mail nur die Verwendung von reinem ASCII-Text, und ist (technologisch) dementsprechend begrenzt. mez und antiorp haben jedoch jeweils eigene Sprachen bzw. Schreibstile entwickelt: mez nennt ihren Stil „M[ez]ang.elle“, während Netochka Nezvanova ihre bzw. Integer seine Sprache „Kroperom“ bzw. „KROP3ROM|A9FF“ nennt. Bei beiden handelt es sich um künstlerische Appropriationen von Programmiercode. Der Programmiersprachen-Unkundige kann in dieser zeitgenössischen „Mailart“, die den Anschein erweckt, als ob eine Datei von einer Software fehlerhaft und unleserlich formatiert oder dekodiert worden wäre, nicht viel mehr als unverständliches Gebrabbel bzw. Datenmüll erkennen. Denjenigen, die sich auf dem Gebiet von Quellcodes und Programmiersprachen als halb-literat erweisen, wird dagegen durchaus klar, dass hier Computercodes und Programmiersprachen verwendet und appropriiert werden bzw. dass auf ihre Ästhetik angespielt wird. Ambivalent bleibt jedoch der Status dieser Sprachen bzw. Sprachfetzen: Er oszilliert in der Wahrnehmung des Rezipienten zwischen unterstellter Ausführbarkeit, also Funktionalität, und Nicht-Ausführbarkeit – Dysfunktionalität – des Code, kurz: zwischen signifikanter Information und asignifikantem Rauschen.

mez' selbsterfundene Kunstsprache „mezangelle“ ist „an Computer-Programmiersprachen geschult [...], ohne aber in strikter Befehlssyntax geschrieben zu sein.“⁶¹ In ihr vermischen

⁵⁹ Sondheim, Alan: Introduction: Codework. In: *American Book Review*, Vol. 22, No. 6, Sept./Okt. 2001, <http://www.litline.org/ABR/Issues/Volume22/Issue6/226sondheim.pdf> [18.11.2003].

⁶⁰ Weitere Pseudonyme sind: data[h!]bleeder, ms post modemism, mezflesque.exe, ova.kill, net.w][ho][urker, Purrsonal Areah Netzwerker.

⁶¹ Cramer, Florian: sub merge {my \$enses. ASCII Art, Rekursion, Lyrik in Programmiersprachen. In: *Text & Kritik*, Themenheft „Digitale Literatur“, hg. v. H. L. Arnold und R. Simanowski, H. 152, Oktober 2001.

sich ASCII Art und Pseudo-Programmcode. Florian Cramer schreibt zu mezangelle: „Wie die portmanteau words von Lewis Carroll und James Joyce verschachteln sich die Wörter der mezangelle doppel- und mehrdeutig ineinander. Die rechteckigen Klammern sind Programmiersprachen und gängigen Notationen der Booleschen Algebra entlehnt, in denen sie mehrere Zeichen gleichzeitig referenzieren, also Vieldeutigkeit beschreiben.“⁶² mez lässt in ihrer Sprache die Texte bzw. Worte physisch zu Kreuzungspunkten unterschiedlicher Bedeutungen werden – wir haben es hier sozusagen mit einer materiell in den linearen Text implementierten Polysemie zu tun. McKenzie Wark beschreibt dies als eine gleichzeitige Anwesenheit der vielen verschiedenen möglichen Potentialitäten in ein und demselben Wort: „Mez introduces the hypertext principle of multiplicity into the word itself. Rather than produce alternative trajectories through the text on the hypertext principle of ‚choice‘, here they co-exist within the same textual space.“⁶³ mez’ Texte treten in ein unendliches Schillern von Bedeutungen ein, die prinzipiell nicht festlegbar sind. Diese Polysemie ist, und darauf weist Florian Cramer hin, wie in vielen Texten von mez, auch eine der Geschlechter: „fe[male]tus’ liest sich simultan als ‚Fötus‘, ‚weiblich‘ und ‚männlich‘. Andere Wörter greifen die Syntax von Dateinamen und Verzeichnisbäumen sowie die Zitierkonventionen von E-Mail und Chats auf.“⁶⁴

„M @ z k ! n 3 n . k u n z t . m2cht . fr3!“ bedeutet – einige werden es vielleicht entziffert haben – „Maschinenkunst macht frei“ und ist der Signatur der Netzentität Netochka Nezvanova a.k.a. Integer entnommen. Integer ist seit 1998 dadurch bekannt geworden, dass sie Mailinglisten mit e-mails bombardiert, die auf den ersten Blick unlesbaren Datenmüll zu enthalten scheinen, also bewusst eine Art von Rauschen in die menschliche Telekommunikation einführt. Auf den zweiten Blick zeigt sich jedoch auch hier, dass es sich um eine Mischung aus menschlicher und Maschinensprache handelt. „Kroperom“ nennt Integer seine Sprache. Die Besonderheit dieser Sprache besteht darin, dass in ihr das phonetische System des lateinischen Alphabets durch die 256 Zeichen des American Code for Information Interchange (ASCII) ersetzt wird, der lingua franca der Computerkultur. Zum Beispiel ist in dem Kroperom-Wort „m9nd“ der phonetische Wert „ine“ durch „nine“ ersetzt worden. Es gibt in diesem Ersetzungssystem jedoch nicht nur phonetische Substitutionen. In „m@zk!n3n kuzt“ ersetzt das „@“ das „a“, das „zk“ den Laut „sch“, das Ausrufungszeichen das „i“ und die Zahl „3“ das „e“. Ersetzungen finden also auch aufgrund visueller

⁶² Cramer: sub merge {my \$enses.

⁶³ McKenzie Wark. Codework. In: *American Book Review*, Vol. 22, Issue 6 (September/October 2001), hier: 5.

⁶⁴ Cramer: sub merge {my \$enses.

Ähnlichkeiten statt (! für ein „i“), bzw. aufgrund visueller und phonetischer Analogien („3“ / three für „e“). Die menschliche Sprache – in diesem Fall eine Mischung aus deutsch und englisch – wird also von Ziffern und Computercode-Metaphern durchsetzt und infiltriert. Außerdem bekommt der Kroperom-Text durch das massive Auftreten von Ausrufungszeichen (das daher rührt, dass der Buchstabe „i“ im Deutschen so oft auftaucht) eine emphatische Qualität und verwandelt die ausführbaren Computerbefehle in oppressive, manchmal auch komische, menschliche Befehle und Aufforderungen: „Tu dies! Tu das!“ Der Leser muss unterschiedliche Strategien einsetzen, um das aus alphabetischen Buchstaben, Zahlen und ASCII-Zeichen bestehende Skript zu entziffern. Dies erschwert und destabilisiert den Leseprozess und löst verschiedenste Assoziationen aus. Josephine Berry schreibt dazu: „The act of reading becomes a pointedly self-reflexive and, in terms of chaos theory, nonlinear experience with each word representing a junction of multiple systems.”⁶⁵ Es bleibt offen, ob der Kroperom-Text, der sehr stark einem ausführbaren Programmcode ähnelt, an einem anderen Ort im Computer tatsächlich kompilierbar, und in der Folge maschinenlesbar, lauffähig, bzw. ausführbar wäre.

```

/*UG monster_army (1 0 0) (-16 -16 -24) (16 16 40) Ambush
*/
void() monster_army =
{
  if (deathmatch)
  {
    remove(self);
    return;
  }
  precache_model ("progs/soldier.mdl");
  precache_model ("progs/h_guard.mdl");
  precache_model ("progs/gib1.mdl");
  precache_model ("progs/gib2.mdl");
  precache_model ("progs/gib3.mdl");

  precache_sound ("soldier/death1.wav");
  precache_sound ("soldier/idle.wav");
  precache_sound ("soldier/pain1.wav");
  precache_sound ("soldier/pain2.wav");
  precache_sound ("soldier/sattck1.wav");
  precache_sound ("soldier/sight1.wav");

  precache_sound ("player/udeath.wav");          // gib death

  self.solid = SOLID_SLIDEBOX;
  self.movetype = MOVETYPE_STEP;

  setmodel (self, "progs/soldier.mdl");

  setsize (self, '-16 -16 -24', '16 16 40');
  self.health = 30;

  self.th_stand = army_stand1;
  self.th_walk = army_walk1;
  self.th_run = army_run1;
  self.th_missile = army_atk1;
  self.th_pain = army_pain;
  self.th_die = army_die;

  walkmonster_start ();
};

```

Auszug aus Jodi: sound (self, CHAN_VOICE, "soldier/idle.wav", 1, ATTN_IDLE); walkmonster_start (); In: *Nettime*, 1.1.1904.

Ob es sich um ausführbaren Code handelt bleibt auch in Jodis „walkmonster_start ()“-e-mail fraglich. Es ist vielleicht eher das Wissen um die potentielle Ausführbarkeit und

⁶⁵ Berry, Josephine: *The Thematics of Site Specific Art on the Net*. Dissertation, University of Manchester. September 2001 [unveröffentlichtes Typoskript].

Performativität, die hier eine Rolle spielt, nicht so sehr die Ausführbarkeit selbst. Bezüglich der „Codeworks“ ließe sich erneut die Frage stellen (die vielleicht auch meine Verwendung des Performativitätsbegriffs⁶⁶ aufgeworfen hat): Hat formaler Programmcode ein Publikum außer der Maschine, die er adressiert? Kann formaler Code ohne die Maschine, die ihn umsetzt und ausführt, performativ sein? Ich würde behaupten: ja. Es gibt Menschen, die den Code ohne Maschine lesen können. Die haben bei Eintreffen der e-mail von Jodi das gesamte Kriegsszenario von `“walkmonster_start ()“` durchspielen können, ohne es zuvor kompiliert zu haben. Auch wenn diese Antwort eine Behauptung bleiben muss, so wird an dieser Stelle doch eines klar: Das, was auf den ersten Blick wie Rauschen erscheint, ist an anderer Stelle womöglich gar keines. Abhängig vom Kontext werden sinnlose Zeichenfolgen plötzlich zu ausführbaren Befehlen oder, vice versa, performative Programmiercodes zu redundantem Datenmüll.

Analog der häretischen Technik der Glossolalie oder der surrealistischen „écriture automatique“, beides Techniken, die durch Ausschaltung des Bewusstseins (Trance, Schlafzustände) einer göttlichen Instanz bzw. dem Unbewussten das Wort erteilen wollten, wird in „mezangelle“ und „Kroperom“ die menschliche Sprache von maschinensprachlichen SteuerCodes und Algorithmen durchsetzt. Im Gegensatz zur surrealistischen These, dass die Befreiung des Unbewussten zu einer gesellschaftlichen Revolution führen würde, erscheint das Hervorbringen einer Zwitter-Sprache, die halb menschlich, halb maschinell ist, manchmal lustvoll, teils fast zwanghaft zu sein. Wenn wir es hier schon nicht mit den Zeichen einer Machtergreifung zu tun haben, so verweist das konvulsivische Sichtbarwerden des „Postoptisch-Unbewussten“ einmal mehr darauf, dass wir auch hier nicht selber sprechen, sondern im Lacanschen Sinne gesprochen werden.

Die Privilegierung des Programmcodes gegenüber den Oberflächen, der Poiesis gegenüber der Aisthesis führt in den ASCII-Arbeiten bzw. „Codeworks“ von mez, Jodi und Netochka Nezvanova zu einer Befreiung insofern, als diese Fokussierung auf das „Postoptisch-Unbewusste“ eine Ent-Täuschung ermöglicht. Eine Ent-Täuschung darüber, bzw. Verabschiedung des Glaubens z.B. daran, dass auch heutzutage nur dann, wenn eine Kamera anwesend ist, Überwachung stattfindet. Die „Codeworks“ lenken unsere Aufmerksamkeit auf die zunehmende Codiertheit und Programmiertheit unserer medialen

⁶⁶ Ein möglicher Einwand könnte sein: Performativität bezieht sich nur auf menschliches Sprechen, auf Sprechen, das von Menschen an Menschen gerichtet ist. Formaler Code, so könnte der Einwand weiter lauten, richtet sich aber nur an Maschinen. Ich stimme diesbezüglich mit Florian Cramer überein, der bestreitet, dass „Maschinensprache nur von Maschinen lesbar“ ist: „It is important to keep in mind that computer code, and computer programs, are not machine creations and machines talking to themselves, but written by humans.“ (Cramer, Florian: Digital Code and Literary Text, [Typoskript], *Poesis*-Symposium 2001, S. 4f.).

Umgebung. Sie bedienen sich einer „armen“ Schrift im Medium e-mail, die aber gleichzeitig im Kontext der Kommandozeile performativ bzw. ausführbar (*executable*) ist. Indem sie genau mit dieser Ambivalenz von Simplizität und Totalität der Ausführung arbeiten, verweisen sie auf die potentiell totalitäre Dimension des algorithmischen Genotextes. „Software is mind control. Come and get some.“