ZEHN THESEN ZUR SOFTWAREKUNST

FLORIAN CRAMER

1. WORUM ES NICHT GEHT

„Software Art“, wie sie in der freien Netz-Enzyklopädie Wikipedia (Stand: September 2003) definiert wird:

> „Software art is a term for the graphic design of visual elements contained in software, eg. GUI (Graphic User Interface), Icons, etc.“[1]

2. WORUM ES GEHT

Künstler, die mit digitaler Software Werke schaffen, die selbst digitale Datensätze sind, gestalten, wie zuvor nur Schriftsteller, Werke aus Symbolen mit einem Instrumentarium, das selbst nur aus Symbolen besteht. So wie kein Schriftsteller Sprache nur als behelfsmäßiges Mittel zum Zweck eines Kunstwerks nutzen kann, das selbst nicht Sprache wäre, Literatur also allein durch sich selbst, wie in einer rekursiven Schleife, an ihrem Instrumentarium mitschreibt, treten auch die Nullen und Einsen digitaler Kunst in ein intimes Verhältnis zu den Nullen und Einsen der Instrumente, mit denen sie nicht nur gestaltet, sondern auch angezeigt und vervielfältigt werden.

3. OHNE SOFTWARE KEINE DIGITALE KUNST

Jede Annahme ist naiv, daß es Schrift, Bild, Ton, Vernetzung im Computer gäbe, die für sich allein oder auch in „multimedialer“ Kombination zur Verfügung stünden, denn keine dieser Datenformen existiert ohne Computerprogramme, die sie erst als solche hervorbringen; und zwar nicht nur (in der Form z.B. von Text-, Graphik-, Musikbearbeitungsprogrammen) bei ihrer Gestaltung und Bearbeitung, sondern schon bei der bloßen Anzeige (in Browser-, Betrachter-, und Abspielsoftware) und Vervielfältigung (durch Netzwerk- und Betriebssystemsoftware). Jedes digitale Kunstwerk, das nicht selbst Computerprogramm ist, existiert immer nur innerhalb des Rahmens, das vorgefertigte Software ihm definiert. So ist jede digitale Kunst Softwarekunst zumindest in dem weiteren Sinne, daß sie softwaregestützte Kunst ist. Zu Softwarekunst im engeren Sinne wird sie, so mein Vorschlag, wenn sie Software nicht als ein ausgelagertes Hilfsmittel, sondern als Teil ihrer Ästhetik begreift.

Date: 23.9.2003.

[1] `http://www.wikipedia.org/wiki/Software_art`

4. SOFTWAREKUNST MUSS NICHT DIGITAL ODER ELEKTRONISCH SEIN

Ein Computerprogramm ist eine Reihe formaler (algorithmischer) Anweisungen, die von einer Maschine ausgeführt werden kann, aber nicht muß. Wie zum Beispiel:

```
// Classic.walk

 Repeat

 {

 1 st street left
 2 nd street right
 2 nd street left

 }
   2
```

Dies ein Beispiel-Programm für „.walk“ von `http://www.socialfiction.org`, einen „psychogeographischen Computer“, der aus den Straßen von Großstädten konstruiert ist statt aus Transistorgattern und der seine Programme ausführt, indem er Spaziergänger statt Elektronen durch sie leitet. Damit verweist .walk auf zwei historische Vorläufer: Erstens Fluxus- und Konzeptkunst mit ihren para-algorithmischen, minimalistischen Aktionspartituren (wie sie nach dem Vorbild von John Cage unter anderem von George Brecht, La Monte Young und Sol LeWitt verfasst wurden), zweitens auf die Geschichte des modernen Computers als anfangs nur imaginärem, theoretischem Apparat in Gestalt der Turing-Maschine.

5. SOFTWAREKUNST IST NICHT SYNONYM MIT KONZEPTKUNST

Von einer Aktionspartitur wie George Brechts erstem „Lamp Event“ von 1961, das aus der binären Anweisung „on. off.“ besteht,[3] unterscheidet sich .walk insofern, als es eine eingeübte kulturelle Praxis der Nutzung von Computern, Software und ihrer Programmierung reflektiert. Während das „Lamp Event“ künstlerische Software-Programmierung formal antizipiert, weist sich .walk schon seinem Titel nach, der auf Microsofts „.NET“ anspielt, als Teil einer Softwarekultur aus. Somit verweist in ihm nicht Konzeptkunst auf Software, sondern umgekehrt Software zurück auf den Konzeptaktionismus der 1960er Jahre, zu dem auch die Psychogeographie der Situationistischen Internationale gehört, und liest ihn als Computersoftware neu. Dieser Rückgriff selbst jedoch ist nicht mehr konzeptkünstlerisch, sondern historistisch, collagierend, ironisch.

Genau darin widerspricht heutige Softwarekunst jener Gleichung von Kunst und Software, die 1970 mit Jack Burnhams Konzeptkunst-Ausstellung „Software“ im New Yorker Jewish Museum und der Erstausgabe der Videokunst-Zeitschrift „Radical Software“ aufgestellt wurde.[4] Software ist, dreißig Jahre später, nicht mehr Laborkonstrukt und Paradigma

---

[2] socialfiction.org, .walk for dummies, `http://www.socialfiction.org/dotwalk/dummies.html`

[3] Partitur-Kärtchen in [Bre64]

[4] Zur Ausstellung siehe [Sha], „Radical Software“ liegt seit kurzem faksimiliert auf `http://www.radicalsoftware.org` vor.

konzeptualistischer Purifikation, sondern seit der Verbreitung von PC und Internet fehlerbehafteter Code, Verursacher von Abstürzen, Inkompatibilitäten, Viren, von Kontingenz also statt Stringenz der Symbole.

Weil die Net.art von jodi, Alexei Shulgin, Vuk Cosic und I/O/D und anderen genau diese Kontingenzen ästhetisierte und damit die Digitalkunst von ihrer akademischen und industriellen Schein-Glätte befreiten, ist es kein Zufall, daß neuere Softwarekunst in einer diskursiven und auch personellen Kontinuität der Netzkunst der 1990er Jahre steht. An der Entwicklung von jodis künstlerischer Arbeit von 1996 bis heute ist beispielhaft ablesbar, wie aus Net.art-Experimenten mit Bildschirmgraphik und Netzwerkkommunikation zunächst eine Arbeit gegen die Grenzen ihrer Softwareumgebung wurde (etwa in der Browser-Manipulation „OSS“ `http://oss.jodi.org`), dann Umprogrammierung von Software (z.B. in dem auf „Quake“ basierenden „Untitled Game“ `http://www.untitled-game.org`) und schließlich eine Reduktion des sichtbaren Objekts auf simple BASIC-Quelltexte (in der neuesten Arbeit „10 Programs written in BASIC ©1984“[5]. Neuere Softwarekunst ähnelt älterer Konzeptkunst äußerlich zwar dann, wenn sie sich minimalistischer Formsprache bedient. Doch ist diese Annäherung widersprüchlich, weil sie nicht im Geiste jener Dematerialisierung des Kunstwerks geschieht, wie sie Lucy Lippard in ihrem Buch „Six Years“ für die Kunst von 1966 bis 1971 insgesamt diagnostiziert, sondern im Gegenteil Software in der heutigen Softwarekunst als Material begriffen wird. Dieses Verständnis ist Voraussetzung auch der schriftkünstlerischen „Codeworks“ u.a. von jodi, antiorp, mez, Alan Sondheim, Johan Meskens und Lanny Quarles,[6] die syntaktische Elemente von Programmiersprachen, Netzwerkprotokollen, Systemmeldungen, Chat-Slang und Umgangssprache vermischen, wie zum Beispiel die folgende E-Mail der Französin Pascale Gustin:

```
L'_eN(g)Rage \ment politi][~isch][K et l' _art is T(od)

][ref lex][1/O.ns 10verses NOT es][

--------\B(L)ien-sUr 2 que/S\tions f.Ond(ent)

---------------------------------------A:

-------------------------------][menta les_sel][l] a tenement) T nem T

-tout d_abord--------1/O(f.ne

1 of 1 deletions
1 deletion done
apply: Command attempted to use minibuffer while in minibuffer
```

6. SOFTWAREKUNST IST NICHT SYNONYM MIT ALGORITHMISCHER KUNST

Wenn Software, allgemein definiert, Algorithmen ist, ist Softwarekunst also gleichbedeutend mit algorithmischer bzw. generativer Kunst? Von Philip Galanter stammt die folgende, hilfreiche Definition generativer Kunst:

---

[5] Ausgestellt bei *Electrohype* in Malmö

[6] Siehe hierzu u.a. [Son01] und [War01]

„Generative art refers to any art practice where the artist creates a process, such as a set of natural language rules, a computer program, a machine, or other mechanism, which is then set into motion with some degree of autonomy contributing to or resulting in a completed work of art.“[7]

Eine Autonomie des Ablaufs, wie sie auch in Jack Burnhams kybernetisch und systemtheoretisch geprägten Aufsätzen der 1960er Jahre beschrieben wird,[8] kann es zwar in Softwarekunst geben; etwa als *running code* in Gestalt klassischer PC-Anwendungssoftware, oder auch als unzweideutige formale Anweisungen wie in „.walk“. Betrachtet man aber populäre Softwarekunst-Subgenres wie zum Beispiel Spielmodifikationen[9] und experimentelle Browser [10], so geht es in ihnen eben nicht um ästhetische Autonomie algorithmischer Prozesse, sondern um deren Bruch durch irritative Verkoppelung von Software, Spielern und Netzwerkdaten. Auch ist in generativer Kunst, nach Galanters Definition, Software nur eines mehrerer möglicher Mittel, das, statt selbst Kunstwerk zu sein, auch bloß zu ihm „beitragen“ kann, so also, wie viele computergestützte Künste (einschließlich elektronischer Musik) Software nicht als Teil ihrer Ästhetik begreifen, sondern im Hintergrund agieren lassen.

Umgekehrt verfehlt Softwarekunst ihrerseits das Kriterium des generativen, oder erfüllt es nur im metaphorischen, nicht im technischen Sinn, dann etwa, wenn sie wie in „Codeworks“ dysfunktionale und imaginäre Software schreibt.

7. Softwarekunst ensteht nicht im Vakuum, sondern als Teil einer Softwarekultur

Wenn also neuere Softwarekunst Software weniger als generative Prozeßsteuerung auffaßt, denn als Spielmaterial, liest sie sie auch nicht mehr, wie in klassischer konzeptueller und generativer Kunst, als reine Syntax, sondern als etwas semantisches, das ästhetisch, kulturell und auch politisch besetzt ist.[11] War Softwarekultur 1970 – wie von Burnhams „Software“-Ausstellung mit ihrer Konfrontation von Konzeptkunst und Forschungslabor-Softwareentwicklung dokumentiert – eine akademische Angelegenheit und auch Hackertum noch auf Eliteinstitute des MIT und in Berkeley beschränkt, existiert heute nicht nur eine Massenkultur und Alltagsästhetik der Software. Wie zum Beispiel die Debatten um Freie Software, Softwaremonopole, Logikpatente oder Nutzer-Ausspionierungs-Programme zeigen, ist Software auch zunehmend zum Politikum geworden. Eine Kulturkritik der Software existiert trotzdem erst in verstreuten Ansätzen, zum Beispiel in den

---

[7] Zitat u.a. auf `http://www.philipgalanter.com/pages/acad/idx_top.html` und `http://www.generative.net`

[8] S.a. die als „Kunst und Strukturalismus“ unglücklich übersetzte deutsche Ausgabe von Burnhams „Structure of Art“, [Bur71]

[9] jodis „Untitled Game“, Joan Leandres „retroyou“ `http://www.retroyou.org`

[10] I/O/Ds „Web Stalker“ `http://www.backspace.org/iod/`, Netochka Nezvanovas „Nebula M.81“, Jodis „wrongbrowsers“ `http://www.wrongbrowser.org`, Mark Napiers „Shredder“ `http://www.potatoland.org/shredder/`, Kensuke Sembos und Yae Akaivas „Discoder“ `http://www.exonemo.com/DISCODER/indexE.html`, Peter Luinings „ZNC Browser“ `http://znc.ctrlaltdel.org/pc_znc2.0.htm`

[11] Der „Injunction Generator“ von Ubermorgen.com `http://www.ipnic.org/intro.html`, der automatisch juristische Abmahnungen erzeugt, und der textzensierenden Web-Proxy-Server „insert coin“ von Alvar Freude und Dragan Espenschied `http://odem.org/insert_coin/` sind zwei überzeugende Beispiele politaktivistischer Softwarekunst.

Aufsätzen von Wolfgang Hagen, Matthew Fuller sowie auf der von Jeremy Hunsinger initiierten Mailingliste „softwareandculture“.[12]

8. SOFTWAREKUNST IST KEINE PROGRAMMIERER-KUNST

Der Graben zwischen „Nutzung“ und „Programmierung“ von Computern ist historisches Resultat der ikonischen Bedienoberfläche und ihrer Kommerzialisierung durch Apple und Microsoft, die beide Operationsweisen erstmals verschiedene Medien zuordneten, ikonische Bilder der „Nutzung“ und alphanumerischen Text der „Programmierung“. Dadurch erst wurde die Programmierung von Computern zur schwarzen Kunst und zum vermeintlich elitären Spezialwissen verklärt.[13] Programmierer haben diesen Mythos ihrerseits kultiviert und das ideologische Erbe des späten 18. Jahrhunderts angetreten, indem sie mit dem Hacker einen Wiedergänger des romantischen Genies kreierten.

So birgt jeder Diskurs über Softwarekunst Gefahr, seinerseits Programmierer-Geniekult zu betreiben. Dem entgegen stehen imaginäre, simulierte und dysfunktionale Software sowie simple, von Laien durchführbare Manipulationen bestehender Software.[14] Wenn Software nicht nur Werkstoff von Softwarekunst sein kann, sondern auch ihr Reflexionsgegenstand, so kann diese Reflexion darüber hinaus in völlig andere Materialien als Software gefaßt werden, wie es z.B. die auf dem „browserday“-Festival 2001 präsentierte Arbeit „n:info“ von Julia Guther und Jakob Lehr demonstriert, ein Browser in der Form eines tragbaren Fensterrahmens, der die Rhetorik ikonischer PC-Software exakt umkehrt, indem er ein analoges Werkzeug als Metapher einer digitalen Software ausweist und damit die Softwareanwendung des Web-Browsing als Kulturtechnik, Wahrnehmungs- und Denkweise exponiert.[15] Nichts spricht also gegen Softwarekunst in der Form z.B. eines gemalten Bildes.

9. GENRE-SCHABLONEN KÖNNTEN SOFTWAREKUNST UNINTERESSANT WERDEN LASSEN

Die Gefahr, in Stereotypen zu erstarren, existiert selbstredend auch in Kunstformen, die, wie etwa Fluxus, sich nicht durch spezifische Materialien definieren. Dennoch: Softwarekunst würde uninteressant, wenn sich ihr Repertoire in der Wahrnehmung von Kritikern, Kuratoren und Jurys auf experimentelle Web-Browser, Daten-Visualisierungen, modifizierte Computerspiele und Cracker-Code (wie Computerviren und Forkbomben) einengen würde. Ein anderes Problem ist die Assoziation von Softwarekunst mit dem Betrieb „Medienkunst“, die dazu führt, daß künstlerisch interessante Computerprogramme, wie sie z.B. im Umfeld von GNU/Linux und freier Software entstehen, nicht in einschlägige Wettbewerben, Festivals und Ausstellungen gelangen.

---

[12] Wolfgang Hagen, Der Stil der Sourcen, [Hag97], Matthew Fuller, Behind the Blip [Ful03], softwareandculture-Homepage und -Archiv `http://listserv.cddc.vt.edu/mailman/listinfo/softwareandculture`

[13] Obwohl es, um einen Computer in einer der gängigen Sprachen zu programmieren, nicht mehr bedarf als zu wissen, was Variablen, Wiederholungsschleifen und if-then-Bedingungen sind.

[14] wie z.B. der „SCREEN SAVER“ von Ivan Khimin und Eldar Karhalev `http://runme.org/project/+screensaver/`, eine Konfiguration des Windows-Bildschirmschoners zu einer suprematistisch-hypnotischen, schwebenden Quadrat.

[15] `http://myhd.org/ninfo`

10. DER STREIT, OB SOFTWAREKUNST „KUNST“ HEISSEN SOLLE, HAT SOFTWAREKUNST NUR SCHEINBAR ZUM GEGENSTAND

Immer wird die Frage laut, weshalb überhaupt Softwarekunst mit dem Suffix „Kunst“ versehen werde. Die naive Version der Frage hält Software schlicht für Ingenieurtechnik und zweifelt daher ihr an ihrem künstlerischen Wert; eine komplexere Variante moniert, daß abermals einer vielseitigen Kultur verzichtbar das „Kunst“-Attribut als Auslesekriterium übergestülpt werde. Und in der Tat: So, wie zum Beispiel die traditionelle japanische Kultur ohne ein Konzept einer freien im Unterschied zur angewandten Kunst ausgekommen ist, zeigt sich auch in der Softwarekultur, in ihren industriellen ebenso wie in ihren freien Spielarten, ein Verständnis von „Kunst“ im Sinne der alten „ars“, von Kunstfertigkeit. Dank des Einfallsreichtums freier Programmierer gelingt es, wie es der Künstler und Festivalkurator Alexei Shulgin zeigt, im Diskurs Softwarekunst besonders gut, Arbeiten von erklärten Künstlern und erklärten Nichtkünstlern zu mischen.[16] Trotzdem sind Einwände gegen das „Kunst“-Suffix von Softwarekunst letztlich nur ein Vehikel, um das Konzept „Kunst“ als solches in Frage zu stellen.

Eine dritte, raffinierte Version des Einwands formuliert Lev Manovich in seiner Rezension „Don’t Call it Art: Ars Electronica 2003“,[17] wenn er Softwarekunst eine Nicht-Kunst nennt, weil sie ihrer Fokussierung auf ein Material wegen aus dem System „zeitgenössische Kunst“ herausfalle. Dabei differenziert sich auch zeitgenössische Kunst, wie sie auf Galerien, Messen und Museumsausstellungen gehandelt wird, auch in Unterdisziplinen aus, die ihrem Material gegenüber alles andere als neutral ausgerichtet sind: Einerseits großformatige Malerei und Fotokunst für Privatsammler, andererseits (oft videogestützte) akademische Installationskunst, die typischerweise in staatlich subventionierten Räumen ausgestellt und von kulturwissenschaftlich trainierten Kuratoren und Künstlern produziert wird. Abgesehen davon, ist Softwarekunst schlicht ein generischer Terminus nicht anders als Malerei, Ton-, Schrift- oder Videokunst, der zudem nicht von Künstlern, sondern von Kritikern und Kuratoren definiert wurde, die in zeitgenössischer digitaler Kunst eine Tendenz hin zur Arbeit mit Software als Material feststellten.[18]

Der Begriff „Softwarekunst“ ist deshalb sehr einfach zu legitimieren, denn er ergibt sich schlicht aus der Tatsache, daß bemerkenswerte zeitgenössische Kunst (so fast alle in diesem Text verzeichneten Arbeiten) in Form von Software entsteht und deshalb eine Theorie und Kritik von Softwarekunst erfordert.

[16] Beispiele dafür sind das auf dem readme-Festival 2002 prämierte Hackerprogramm „WinGluk Builder“ http://www.macros-center.ru/read_me/art_work/27/readme27.zip und das im folgenden Jahr ausgestellte Programm „Tempest for Eliza“ http://www.erikyyy.de/tempest/, das einen einfachen Kurzwellen-Radiosender durch Bildschirmgraphik auf Röhrenmonitoren implementiert.

[17] Publiziert auf den Mailinglisten „Rhizome“ und „Nettime“, [Man03]

[18] So z.B. 1999 Saul Albert in seinem Aufsatz „Artware“ [Alb99] und Alex Galloway in „Year in Review: State of net.art 99“ http://switch.sjsu.edu/web/v5n3/D-1.html, Andreas Broeckmann, der im Jahr 2000 dem Transmediale-Festival eine Software-Sektion hinzufügte und, 2001, Tilman Baumgärtel mit seinem Artikel „Experimentelle Software“ [Bau01].

LITERATUR

[Alb99] ALBERT, Saul: Artware. 1999. – http://twenteenthcentury.com/saul/artware.htm 6

[Bau01] BAUMGÄRTEL, Tilman: Experimentelle Software. In: *Telepolis* (2001). – http://www.heise.de/tp/deutsch/inhalt/sa/9908/1.html 6

[Bre64] BRECHT, George. WATER-YAM. 1986 (1964) 2

[Bur71] BURNHAM, Jack: *Kunst und Strukturalismus*. Köln : DuMont, 1973 (1971) 4

[Ful03] FULLER, Matthew (Hrsg.): *Behind the Blip. Essays on the Culture of Software*. Brooklyn : Autonomedia, 2003) 5

[Hag97] HAGEN, Wolfgang: Der Stil der Sourcen. Anmerkungen zur Theorie und Geschichte der Programmiersprachen. In: COY, Wolfgang (Hrsg.) ; THOLEN, Georg C. (Hrsg.) ; WARNKE, Martin (Hrsg.): *Hyperkult*. Basel : Stroemfeld, 1997, S. 33–68 5

[Man03] MANOVICH, Lev: Don't Call It Art: Ars Electronica 2003. 2003. – http://amsterdam.nettime.org/Lists-Archives/nettime-l-0309/msg00102.html 6

[Sha] SHANKEN, Edward A.: The House that Jack Built: Jack Burnham's Concept of 'Software' as a Metaphor of Art. In: *Leonardo Electronic Almanach* 6, Nr. 10. – http://www.duke.edu/~giftwrap/House.html 2

[Son01] SONDHEIM, Alan: Introduction: Codework. In: *American Book Review* 22 (2001), September, Nr. 6, S. 1–4 3

[War01] WARK, McKenzie: Essay: Codework. In: *American Book Review* 22 (2001), September, Nr. 6, S. 1–5 3