WEB STALKER SEEK AARON

Gedanken zu digitaler Kunst, Code und Codierer

Erkki Huhtamo

„[J]ede Form von Software lässt uns unseren historischen Kunstbegriff überdenken."
(Jack Burnham, „Notes on art and information processing", Katalog zur Software-Ausstellung, 1970)

1.

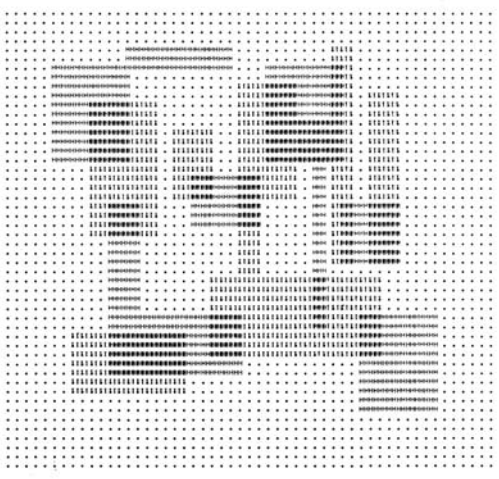
In der Mainstream-Computertechnik zeichnet sich ein Trend zur Verschleierung des Codes ab. Während das Programmieren und Lesen von Code für die ersten Computeranwender – Wissenschaftler, Techniker, Operatoren und sogar Künstler – eine Selbstverständlichkeit war, bleibt der Code heute zunehmend unsichtbar bzw. hinter der Fassade der Benutzeroberfläche verborgen. Seit Ende der 1960er Jahre ist die Vernetzung von Mensch und Computer über „benutzerfreundliche" Interfaces eines der grundlegenden Ziele der digitalen Kultur. Wie Nicholas Negroponte 1969 anmerkte, bestand für ihn die wahre Herausforderung darin, dem Computer ein Verständnis des Menschen zu vermitteln, und nicht vice versa: „Ein Entwickler sollte beim Arbeiten mit einer Maschine nicht auf maschinenorientierte Codes zurückgreifen müssen. Trotz aller Fortschritte in der Computertechnologie gilt als Paradigma für einen sinnvollen Dialog, dass Maschinen auf natürliche Sprache reagieren können."[1] Die Bemühungen „nahtlose" und „intime" Mensch-Computer-Interfaces zu schaffen, wurden zum Symbol einer progressiven Computerkultur, zu der sowohl Experten als auch normale User gehören. Von Xerox Star bis Apple Macintosh und dessen unehelichem Nachkommen, Microsoft Windows, wurden Generationen von Usern darauf gedrillt, die Vorgänge im Inneren der „Kiste" zu ignorieren. Sie bedienten sich einer Software, deren metaphernreicher Inhalt sich per Mausklick öffnen ließ. Statt binärer Ketten von Nullen und Einsen – dem für die Steuerung des Systems erforderlichen Programmcode – sahen die User Icons und Papierkörbe auf dem Desktop.

Während Computer in allen erdenklichen Bereichen zum Einsatz kommen, bleibt der direkte Zugang zum Code und zur Programmierung Spezialisten vorbehalten: Informatikern, Berufsprogrammierern, Hackern und Entwicklern von Game-Engines. Mac- und sogar die meisten Windows-User sehen den Code nur selten hinter dem Interface hervorblitzen. Auch die kurze Euphorie, als mit dem Auftauchen von HTML das Code-Schreiben auch für Durchschnittsuser zur Routine wurde, flaute mit der Entwicklung einer Reihe von einfach zu bedienenden Website-Erstellungsprogrammen rasch ab. Eine mögliche Erklärung für diese Entwicklung ist, dass der Computer nichts anderes als eine Vermittlungsinstanz, ein Werkzeug bzw. Medium ist und nicht das Ziel per se darstellt; es finden sich allerdings auch eine Reihe von Gegenargumenten. Die zunehmend wichtigere Rolle des Computers als universell einsetzbare Maschine, die zahlreiche Anwendungen und Systeme weltweit steuert, lässt die völlige Unsichtbarkeit des Innenlebens von Computern bedenklich erscheinen. Programmierer von internationalen Konzernen und Regierungseinrichtungen (Code-Knacker nicht zu vergessen) ebenso wie Pranksters, Cyber-Kriminelle und Terroristen überwachen und manipulieren via Internet die „unschuldigen" Computeranwender. Da den Usern das nötige Programmierwissen fehlt, können sie nur auf kommerzielle Dienste zurückgreifen oder handelsübliche Softwarepakete installieren. Der eigentliche Kern des Problems bleibt allerdings bestehen und mögliche Gegenmaßnah-

men sind nicht ausreichend durchdacht. Es wird auch argumentiert, dass durch den Einsatz kommerzieller Software die Ausdrucksfreiheit der Anwender eingeschränkt und ihnen ohne ihr Wissen die von den Konzernen vorgesehene Rolle aufgezwungen wird.[2] John Bergers berühmter Slogan „Jedes Bild verkörpert eine bestimmte Sichtweise" könnte in abgewandelter Form lauten: „Jede Software verkörpert einen bestimmten Benutzungsmodus." [3] Zugang zum Code, das Verstehen der „Botschaft" und die Fähigkeit, Code für persönliche Zwecke zu nutzen, sind politische, soziale und wirtschaftliche Fragen, die eng mit den bestehenden Machtverhältnissen und der Wissensdynamik in unserer Gesellschaft verknüpft sind.
Vor diesem Hintergrund scheint die Konzentration auf den Code in der digitalen Kunst überaus interessant. Dieses Interesse zeigt sich auf unterschiedliche Weise, besonders jedoch im neu entstandenen Diskurs über „Softwarekunst".[4] In den letzten Jahren entstanden Kunstwerke, die das Aussehen und die Funktionalität kommerzieller Softwareapplikationen, von Webbrowsern bis zu interaktiven Spielwelten, veränderten und Features einführten, die von den Usern als formale Interventionen, störende Pranks, ideologische Subversion oder einfach technische Bugs interpretiert werden können. Der berühmte *Web Stalker* (I/O/D, 1997-) ist ein Webbrowser, der anstatt des herkömmlichen grafischen Interface Kontrollcodes und Linkstrukturen anzeigt. *Life_Sharing* von 0100101110101101.ORG richtete sich gegen die als falsch empfundene Offenheit von Webbrowsern, indem Usern über das Internet der Zugriff auf die Festplatte einschließlich aller privaten Dateien und Programmen gestattet wurde.[5] Andere Arbeiten setzen Elemente des Computercodes als Gestaltungsmittel ein, ohne diese hinter Grafiken, Bildern und Klängen zu verbergen. All diese Formen der digitalen Kunst versuchen die Konventionen der Computernutzung zu hinterfragen. Indem sie den Schleier der als trügerisch empfundenen „Benutzerfreundlichkeit" lüften, gestatten die Künstler den Usern einen Blick hinter die Kulissen und in den „Maschinenraum" (um eine etwas anachronistische Metapher zu verwenden) zu werfen. Sie fühlen sich in diesem Raum wohl, betrachten ihn als Labor, Treffpunkt, Toolkit und Ort der Kunst. Woher rührt jedoch dieses Interesse? Geht es weit genug, diese Versuche als „unvermeidbare" Avantgardebewegung der digitalen Kultur zu bezeichnen? In welchem Bezug steht dieses Interesse zum weiteren, vielleicht sogar nicht-digitalen, kulturellen Kontext? Dieser Artikel versucht Antworten auf einige dieser einleitenden Fragen zu geben, indem die Beziehung zwischen Kunst, Code und Codieren aus einer medien-archäologischen Perspektive betrachtet wird.

2.

Die ersten Computerkünstler der 1960er Jahre waren Programmierer. Es gab keine Alternative. Jedes Kunstwerk, ob Grafik, Animation oder Musik, war notwendigerweise das Ergebnis eines einzigartigen Codierungsprozesses. Als ob sie die aktuellen Ambitionen der „Softwarekunst" bereits geahnt hätten, erklärten Computerkunst-Pioniere wie Michael L. Noll und die Mitglieder der japanischen Computer Technique Group (CTG) unumwunden, dass das generierende Programm und nicht das computergenerierte Ergebnis das wahre Kunstwerk sei.[6] Die meisten frühen Kunstwerke der Computergrafik und -musik verschrieben sich einem formalistischen Ansatz und konzentrierten sich u. a. auf die Vor- und Nachteile generativer Grammatiken und das Verhältnis zwischen regelbasiertem Verhalten und zufallsbestimmten Ereignissen. Dieser Hintergrund zeigt sich klar in Jasia Reichardts früh erschienener Einführung *The Computer in Art* (1971), die diese Thematik anhand zahlreicher Diagramme anschaulich beschreibt.[7] Computergrafik-Pioniere wie Frieder Nake beschrieben ihr Kunstschaffen als „visuelle Forschung", die bewusst soziale oder politische Anliegen aussparte. Inspiration fanden sie in der Kybernetik, Claude Shannons Informationstheorie und der exakten, mathematischen Ästhetik von Theoretikern wie Max Bense.8 Bense klammerte die Rolle der subjektiven Wahrnehmung aus und basierte sein ästhetisches System auf mathematischen Gleichungen, die



Katherine Nash: ART 1

Harold Cohen: Tate Gallery

das Universelle statt des Besonderen, das Rationale statt des Irrationalen (gleichzusetzen mit dem subjektiven Kunstimpuls), das Abstrakte statt des Repräsentationalen reflektierten. Obwohl die ersten Kunst-Programmierer häufig Zufallsoperationen nutzten, wurde der Computer primär als „Werkzeug" betrachtet: Er sollte ein zuvor geschriebenes Programm ausführen.
Diese Ausgangslage lässt sich sowohl durch den Stand der Technik als auch den weiteren institutionellen Kontext erklären. In den 1960er Jahren wurden Computer meist in Regierungseinrichtungen und Unternehmen verwendet. Sie wurden primär für statistische Berechnungen genutzt, die das Programmieren in erster Linie als Batch-Processing betrachteten. Obwohl die ersten Künstler den Computer für andere Zwecke nutzen wollten, waren sie gezwungen, sich in die von der nicht-künstlerischen „Mainframe"-Kultur vorgegebenen institutionellen Rollen einzufügen: In einschlägigen Institutionen schrieben sie Programme und warteten dann, dass diese vom Computer ausgeführt wurden. Sie waren Mitglieder einer kleinen Elite am Rand einer größeren institutionalisierten Technik-Elite, die ein unglaublich expressives Medium ausloteten, das noch in den Kinderschuhen steckte. Allerdings veränderte sich das Wesen der Computertechnik durch Innovationen wie neue Interfaces, Time-Sharing und die Entwicklung der ersten Programmiersprachen für künstlerische Zwecke kontinuierlich. BEFLIX, ein vom Informatiker Kenneth Knowlton von Bell Labs entwickeltes Programm, wurde von Stan Vanderbeek und Lillian Schwartz in Zusammenarbeit mit Knowlton für innovative Computergrafiken und -animationen eingesetzt. Ebenso interessant war ART 1 der Universitätsprofessoren Katherine Nash und Richard H. Williams, das als Werkzeug für Künstler gedacht war, die nicht über die nötigen theoretischen und technischen Programmierkenntnisse verfügten, aber Computer für ihre Kunstprojekte einsetzen wollten. Obwohl das Programm heftig kritisiert wurde, da der Künstler „als Spezialist mit vordefinierten professionellen Bedürfnissen" betrachtet wurde, war es doch der erste Hinweis darauf, dass sich die Wege der Entwickler von „Soft"-Kunst und reinen Algorithmus-Schreibern bald trennen würden.[9]
Die Verbreitung von Kunst, die eingeführte Software wie Photoshop oder Maya nutzt, hat die Entwicklung neuer eigenständiger Algorithmen nicht verdrängt. Viele der nachhaltigsten digitalen Kunstprojekte wurden von Künstlern initiiert, die ihre Software entweder selbst schreiben oder eng mit einem Programmierer zusammenarbeiten. Beispiele für Erstere sind Myron Krueger, Harold Cohen und David Rokeby, während Künstler wie Jeffrey Shaw (gemeinsam mit Gideon May und Bernt Lintermann) und Rafael Lozano-Hemmer (gemeinsam mit Will Bauer)

der zweiten Gruppe angehören. In manchen Fällen, wie z. B. bei Christa Sommerer und Laurent Mignonneau, ist eine Trennung zwischen Programmieren und anderen Aspekten des Kunstschaffungsprozesses nahezu unmöglich.[10] Pioniere wie Krueger, Cohen und Rokeby haben Jahre, sogar Jahrzehnte, damit verbracht, Programmcode zur Weiterentwicklung ihrer zunehmend komplexen Systeme zu schreiben. Die Werke dieser Künstler haben im Lauf der Jahre ihre eigene Identität entwickelt, können jedoch gleichzeitig als „Materialisationen" dieser Systeme betrachtet werden, die den aktuellen Entwicklungsstand der Programme dokumentieren. Code ist das zentrale Element von Kruegers *Videoplace* und Rokebys *Very Nervous System* oder *Giver of Names.*[11] Harold Cohens *AARON* ist wohl das langfristig konsequenteste Unterfangen, eigenen Programmcode für die Schaffung eines evolutionären Kunstprojekts einzusetzen. Es wurde seit Anfang der 1970er Jahre kontinuierlich weiterentwickelt.[12]
AARON, ein KI-basiertes Computerprogramm, ist ein Expertensystem zur Generierung von Bildern und Zeichnungen. In den letzten dreißig Jahren wurden verschiedenste Ausgabegeräte entwickelt, von einer Zeichen-„Schildkröte", die sich auf am Boden liegenden Papier fortbewegt, bis zu komplexen Zeichenmaschinen und – als neuester Entwicklung – einer Softwareanwendung, die automatisch Bilder am Desktop generiert. AARON kann als halbautonomes System betrachtet werden. Die Zeichnungen basieren auf den von Cohen definierten komplexen Regeln, weisen jedoch, von der Bildkomposition bis zur Farbgebung, einen beträchtlichen Grad an Autonomie auf. Es ist bezeichnend, dass Cohen den Code von AARON nie freigegeben hat. Die verschiedenen Entwicklungsstadien des Programms wurden nicht systematisch dokumentiert, weshalb die Entwicklung des Codes nur indirekt über die zahlreichen von AARON generierten Zeichnungen und Bilder entschlüsselt werden kann.[13] Cohen betrachtet den Code nie als Kunstwerk per se, obwohl eine beachtliche schöpferische Leistung dahinter steckt. Code kann Cohen zufolge vielmehr mit den Fertigkeiten und Techniken, die ein Mensch Zeit seines Lebens erwirbt, verglichen werden. Einzig die Bilder, die materiellen Spuren eines Lebenswerks, haben Bestand. In diesem Sinn bedient sich Cohen, selbst ausgebildeter Maler, eines konventionellen Modells zur Schaffung und Konservierung von Kunst. Die neueste Version von AARON, die kostenlos über das Internet erhältlich ist, könnte jedoch eine etwas radikalere Richtung einschlagen.[14] Anstatt weiter als Cohens persönliche kybernetische Verlängerung zu fungieren, wurde AARON von seinem Schöpfer ein gewisses Maß an Unabhängigkeit zugestanden. Allerdings wurde mit der Freigabe von AARON als frei verfügbare Softwareanwendung auch die Weiterentwicklung von AARON abgebrochen.

Harold Cohen: *Clarissa*

Harold Cohen: *Machine*

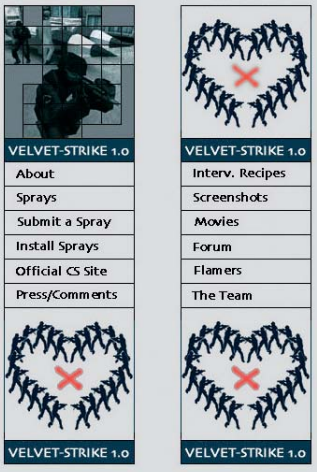
Anne-Marie Schleiner: Velvet-Strike
http://www.opensorcery.net/velvet-strike/

Anne-Marie Schleiner: Velvet-Strike
http://www.opensorcery.net/velvet-strike/

Obwohl weiterhin zahllose unterschiedliche Bilder generiert werden können, kann das Programm keine neuen Routinen erlernen. Als Desktopinstallation verewigt AARON zweifelsohne Cohens Errungenschaften, wird jedoch gleichzeitig auch zu einem kybernetischen Zombie. Mit der Freigabe des Quellcodes bestünde eine Chance, das Programm mittels kollektiven Programmierens im Internet weiter zu entwickeln.[15]

3.

Die Vertreter der „Softwarekunst" weisen auf die Bedeutung des Codes als wichtigste kreative Leistung hin und fordern eine uneingeschränkte Präsenz und Funktion des Codes im Kunstwerk. Laut einer Erklärung der Jury des ersten Software-Kunstwettbewerbs der Transmediale 01 „richtet sich Softwarekunst gegen die Auffassung von Software als Mittel zum Zweck".[16] Für die Jury hat Softwarekunst verschiedene Gesichter: „Softwarekunst könnte Algorithmen als Selbstzweck präsentieren, traditionelle Paradigmen untergraben oder neue hervorrufen, innovative oder störende Prozesse in einem System auslösen, eine Form des kreativen Schreiben oder sogar eine Wissenschaft sein." In einer weiteren Erklärung führen die Jurymitglieder Florian Cramer und Ulrike Gabriel aus: „Softwarekunst bedeutet eine Verschiebung der Perspektive des Künstlers von der Präsentation hin zur Entwicklung von Systemen und Generierung von Prozessen an sich; das ‚Medien' -Konzept deckt dies jedoch nicht ab."[17] Für die oben erwähnten Autoren besteht die „Hauptsünde" der Medienkunst darin, dass dem Design des Interface zu viel Aufmerksamkeit gewidmet wurde. Für den Softwarepuristen ist die Schaffung von komplexen immersiven Umgebungen und vielschichtigen multisensorischen Interfaces ein Akt der Mystifizierung. Kunstwerke, die die Betrachter sowohl körperlich als auch emotional teilhaben lassen, „verführen" diese, anstatt die wahre Natur des Systems, die „hinter der Fassade" verborgen bleibt, erkennen zu lassen.
Es scheint überaus interessant, dass interaktive Systeme, die vor nicht allzu langer Zeit als bedeutsame und kritische Alternative zu „passivierenden" Medien wie dem Fernsehen betrachtet wurden, nun die Zielscheibe der Kritik sind. Diese veränderte Sichtweise beruht auf der zunehmenden Akzeptanz der Interaktivität und ihrer Integration in den Mainstream der digitalen Kultur. Wurde Interaktivität früher als Geste betrachtet, die die vorherrschende Medienlogik in Frage stellte, so gilt sie nun als „sinnvoll" und Teil des internen kulturellen Repertoires. Die jüngste Kritik scheint sich allerdings nicht an die Interaktivität selbst zu richten, sondern vielmehr gegen die Art, wie Interaktivität präsentiert und vermarktet, institutionalisiert und kommodifiziert wird. Im Computerspielbereich wird etwa argumentiert, dass die Quasi-Echtzeitinteraktion zwischen dem User und der Spielsoft-/-hardware zur „Automatisierung" des Reiz-/Reak-



Anne-Marie Schleiner: Velvet-Strike
http://www.opensorcery.net/velvet-strike/

tionsmechanismus führt. Das Eintauchen in sich rasch verändernde emotionale Spielwelten lässt wenig Zeit zur Reflexion der der Spielerfahrung zugrunde liegenden Algorithmen.[18] Für den Spieler scheint das System selbst nicht zu existieren, es zählt lediglich die in der Spielwelt gemachte phänomenologische Erfahrung. Die Entstehung von Phänomenen wie Game-Patch-Art ist insofern interessant, als solche Identifikationsmechanismen mittels Programmieren direkt angesprochen werden. Ein aktuelles Beispiel liefert Anne-Marie Schleiners *Velvet-Strike*-Projekt (2002), in dem Menschen digitale Graffiti auf die Wände von *Counter-Strike,* dem populären Online-Shooter zum Thema Terrorismus, sprühen.[19] Game-Patch-Art, eine Kunstform, die im Spektrum zwischen Spielsubkultur, die in anderen Texten „wildert", und kritischer Softwarekunst angesiedelt ist, fungiert als zweideutiger Gegendiskurs zur kommerziellen Spielkultur und bleibt dem Hackertum treu.

Allerdings zeigt die jüngere Medienkunst auch Anzeichen von Selbstreflexion und wachsendem kritischen Bewusstsein zu ihrer eigenen Position gegenüber kommerziellen und institutionellen Anwendungen. Werke wie Maurice Benayouns und Jean-Baptiste Barrières *World Skin* (1998) oder Ken Feingolds *That Sinking Feeling* (2002) sind weit mehr als naive Lobgesänge auf die Annehmlichkeiten des Interface. Während sie weiterhin unvergessliche interaktive Erfahrungen möglich machen, manipulieren sie gleichzeitig die Beziehung zu den Usern. 20 Beide Werke schaffen zweideutige Situationen, in denen das verführerische Potenzial der Interaktivität stets von diskrepanten *(World Skin)* oder vorsätzlich „defekten" *(That Sinking Feeling)* Elementen unterminiert wird. Obwohl keines dieser Werke sich primär mit dem Computercode auseinandersetzt, findet die Rolle des Codes dennoch Beachtung. Die Algorithmusebene dieser Werke steht zwar nicht im Mittelpunkt, die Funktion der Algorithmen aber in engem Zusammenhang mit zentralen, von diesen Arbeiten thematisierten Fragen, wie der Rolle der Medien, der gegenwärtigen Politik der Simulation, der Sozialpsychologie oder der Identitätskonstruktion. Auf ihre eigene Art demonstrieren sie, dass die Konzentration auf den Code auf Kosten anderer möglicher Perspektiven vielleicht nicht der richtige Weg zu einer integrierten digitalen Kultur ist, in der technische, ideologische und kulturelle Codes unwiderruflich miteinander verflochten sind.

4.

Es ist interessant, dass die Softwarepuristen bereits begonnen haben, ihre eigene Genealogie zu verfolgen. Dies bekräftigt die allseits bekannte Tatsache, dass Geschichte nur geschrieben wird, um die Gegenwart zu rechtfertigen. Ein einschneidendes Ereignis in der Urgeschichte der Softwarekunst war die von Jack Burnham 1970 für das Jewish Museum in New York kuratierte „Software"-Ausstellung. Diese technisch katastrophale und bislang weithin unbekannte Ausstellung kann als Schnittpunkt betrachtet werden, an dem die Anstrengungen zur Erforschung des kreativen Potenzials der Informationstechnologie und Formen der Konzeptkunst ebenso wie Burnhams persönliches Interesse an strukturalistischen Analysen zusammenflossen.[21] In seiner Einleitung zum Ausstellungskatalog stellte Burnham fest, dass

SEEK by Nicholas Negroponte and the Architecture Machine Group at MIT, 1969–70. Shown at "Software", Jewish Museum, New York, 1970.

„Software" keine herkömmliche Kunstausstellung sei. Vielmehr würden Objekte ausgestellt, die sich mit „konzeptuellen und prozessualen Beziehungen" beschäftigten. Eines der Ziele war es, „normale Erwartungen und Wahrnehmungsmuster der Ausstellungsbesucher zu untergraben".[22] Die Besucher sollten mit verschiedenen technischen Objekten interagieren, ohne diese zwangsläufig als Kunstwerke zu betrachten. Das auffallendeste Ausstellungsobjekt war SEEK von Nicholas Negroponte und seinen Kollegen von der *Architectural Machine Group* des MIT.

Bei SEEK handelte es sich ebenfalls um ein KI-inspiriertes Programm, das außer auf einer metaphorischen Ebene wohl kaum sein Ziel erreichte: Lebende Wüstenspringmäuse wurden in einen Glaskäfig mit Aluminiumbausteinen gesetzt; ein computergesteuerter Roboterarm konnte von oben in den Käfig herabgelassen werden. Das System war darauf programmiert, die Bausteine nach einem vorprogrammierten Muster anzuordnen, und sollte „intelligent" auf den von den Wüstenspringmäusen produzierten „Lärm", z. B. das Trippeln ihrer Füßchen auf den Bausteinen etc., reagieren.[23]

Für die Anhänger der Softwarekunst war wohl *Labyrinth,* eine frühe Version von Ted Nelsons Hypertext in der Form eines simplen Netzwerks von Knoten und elektronischen Verbindungen, das inspirierendste Ausstellungsobjekt. Verschiedene andere Ausstellungsobjekte ermöglichten den Besuchern mit technischen Apparaten zu interagieren, und unternahmen wenig Bemühungen fiktive Geschichten rund um diese Konstruktionen zu schmieden oder „Bühnenbilder" zu entwerfen. Mit „Software" schuf Burnham eine Verbindung zwischen Computertechnologie und Konzeptkunst; er inkludierte auch nicht-technische Werke von Künstlern wie John Baldessari, Lawrence Weiner und Douglas Huebler. Aus Burnhams Sicht waren diese Objekte weniger Kunstobjekte, sondern sie thematisierten vielmehr tendenziell Sprachstrukturen und Formen des Informationsaustauschs, die anderen nichtsprachlichen Ausdrucksmitteln zugrunde liegen. Dies stand auch im Einklang mit Burnhams Bemühungen, die den westlichen Kunsttraditionen zugrunde liegenden mythischen Strukturen aufzuzeigen.[24] Konzeptkunst – die in einer weit gefassten Auslegung auch John Cages Kompositionsmethode umfasst –, die von den Fluxus-Künstlern generierten Anweisungen für imaginäre Ereignisse und bestimmte Formen lettristischer Lyrik sind zweifelsohne ein mögliches Umfeld, in dem die Rolle von Code in der Softwarekunst analysiert werden kann.

Man könnte sich allerdings auch auf den Bereich der strukturellen und materialistischen Filmproduktion und die Ideologie des „Anti-Illusionismus" in der Filmkultur der späten 1960er und frühen 1970er Jahre konzentrieren. Als Kritik am Illusionismus des konventionellen narrativen Kinofilms begannen Filmemacher das Kino in all seinen prägenden Elementen zu dekonstruieren. Sie betonten die Materialität des Films, inklusive der Klebestellen, Einzelbilder, Emulsionen, Kratzer und Schmutzpartikel. Manche Filmemacher, darunter Hollis Frampton, Michael Snow oder der Kroate Ladislav Galeta, verwendeten (quasi-)generative Prinzipien zur Strukturierung ihrer Filme. In den extremsten Projekten dieser Bewegung verzichteten die Filmemacher gänzlich auf den Film und inszenierten Aktionen, die sich auf die grundlegenden Elemente des Kinos konzentrierten: den Lichtstrahl des Projektors, die Leinwand, die Dunkelheit im Saal. In einem Beitrag in der richtungweisenden *Structural Film Anthology* (1976) definierte Peter Gidal strukturalistisch/materialistische Filme als „Objekt und Prozess zugleich".[25] In einem ande-

ren Aufsatz stellt er fest: „Ein Film ist materialistisch, wenn er den ihm zugrunde liegenden illusionistischen Apparat nicht verbirgt. Es handelt sich hier nicht um reinen und simplen Anti-Illusionismus, um das Aufdecken von Wahrheiten, sondern um einen kontinuierlichen Prozess gegen die Schaffung einer illusionistischen Hegemonie.“[26] In anderen Worten, der materialistische Film war nicht so sehr ein Reinigungsritual, als vielmehr ein beständiger Kampf gegen hegemoniale Kräfte, die sich des Illusionismus bedienten.

5.

Die Bemühungen der strukturalistischen und materialistischen Filmbewegung könnten mit den Ambitionen der Softwarekünstler verglichen werden, die gegenwärtig versuchen, den glatten Fassaden der Cyberkultur, die sich metaphorisch in einer trügerischen Offenheit und einer fingierten Demokratie des grafischen Userinterface manifestieren, Kratzer zuzufügen.[27] Vergleiche über die Zeiten hinweg sind allerdings gefährlich. Es ist nur beschränkt zulässig, Parallelen zwischen einem computergenerierten Bild der 1960er Jahren, das dicht mit ASCII-Zeichen übersät war, und einem Objekt der „ASCII-Kunst“ der späten 1990er Jahre herzustellen. Jene Hacker, die als Studenten *Spacewar,* das als das erste Computerspiel gilt, entwickelten, haben nur wenig mit professionellen Spielentwicklern oder Game-Patch-Künstlern von heute gemein. Ähnlich können die Programme der Computergrafik-Pioniere der 1960er aufgrund des unterschiedlichen kulturellen Umfelds nicht direkt mit der Softwarekunst des frühen 21. Jahrhunderts verglichen werden. Der Diskurs über Softwarekunst hat sich in einem Kontext entwickelt, in dem die digitale Kultur bereits genügend Zeit hatte, Geschichte zu schreiben und Erinnerungen zu entwickeln. Während der ersten 50 Jahre durchlief die digitale Computertechnik zahlreiche Entwicklungsstadien, die die Bedeutung von Computern in verschiedenen Bereichen (Kriegsführung, Verwaltung, Gesellschaft, Wirtschaft und Kultur) neu definierten. Phänomene wie künstliche Intelligenz, virtuelle Realität, Agenten und Avatare, A-Life-Systeme, GUI-Design, Physical Computing und digitales Networking sind Phänomene eines sich rasch entwickelnden Systems, das je nach Zeit und Ort bzw. Position und Identität des Beobachters seine Form ändert. Nicht zuletzt vermag die Softwarekunst von den Erfahrungen der Netzkunst-Pioniere zu profitieren.

Hinsichtlich der Geschichte der Digitaltechnik beweist die digitale Kunst in den letzten Jahren eine gewisse Reflektiertheit. Das aktuelle Interesse der Künstler an KI bedeutet jedoch kein künstlerisches Wiederaufleben der klassischen KI-Forschung. Es handelt sich auch nicht um eine simple Hommage. Projekte wie David Rokebys *Giver of Names,* Kenneth Rinaldos *Autopoiesis* und Ken Feingolds sprechende und interagierende Marionetten sind Beispiele einer Metakunst, die den Dialog mit den kulturellen Repräsentationen der KI aufnimmt (wie im Fall Feingolds mit Joseph Weizenbaums quasi-intelligentem Konversationsprogramm ELIZA), während gleichzeitig andere intellektuelle und ideologische Ziele verfolgt werden. Es ist vielleicht nicht falsch zu behaupten, dass es eine kulturelle Nachfrage nach einem Phänomen wie Softwarekunst gibt, so wie eine Nachfrage nach strukturalistisch/materialistischen Filmen bestand; diese Nachfrage ging von einem Konglomerat von widersprüchlichen kulturellen Kräften aus, von der wachsenden Uniformität und „Unberührbarkeit“ der kommerziellen Filmproduktion bis zum Einfluss kultureller Gegenbewegungen, dem Entstehen dekonstruktionistischer Philosophien und der „Dematerialisierung des Kunstobjekts“. Die strukturalistisch/materialistische Filmbewegung entwickelte sich als kritische Gegenbewegung zur dominanten audiovisuellen Hegemonie und forderte einen Zugang, der die „Primitiva“ des Mediums Film in einem dynamischen Wechselspiel mit seiner Anwendung für narrative und metamorphe Zwecke aufzeigt. In ihrer anti-illusionistischen Rigidität erweckte sie den Anschein einer modernistischen Gegenbewegung, die jedoch nicht naiv-revivalistisch war.

Die von den Befürwortern der Softwarekunst aufgestellten Forderungen und Prognosen besit-

zen jedoch einen gewissen neo-modernistischen Anstrich. Die Betonung der zentralen Funktion des Codes und der Algorithmusebene symbolisiert die Konzentration auf einen „harten Kern", der in der postmodernen Welt oft verschwunden geglaubt wurde. Es gibt tatsächlich „kleine, jedoch wenig einflussreiche" Gruppen (um es unverblümt mit Vuc Cosic' Worten auszudrücken) – wie jene um Geoff Cox, Alex McLean, Adrian Ward u. a., die sich der Erforschung der Ästhetik von generativem Code widmet –, die viele Merkmale einer klassischen Avantgarde-Bewegungen aufweisen.[28] Florian Cramer beschrieb die Aktivitäten dieser Gruppe, wie z. B. Lesungen von Perl-Script-Gedichten, als „Softwareformalismus".[29] Andererseits finden sich Gruppen, die die kulturellen und ideologischen Grundfesten des Programmierens akzentuieren. Für die britische *Mongrel*-Gruppe oder für I/O/D (Erfinder des *Web Stalker)* kann der digitale Code nicht von den verschiedenen Manifestationen der Ideologie im Internet bzw. anderen Umgebungen losgelöst werden. Ihre Aktionen und Projekte lassen sich schwer in eine modernistische „Zwangsjacke" stecken. Noch komplexer wird die Situation bei unabhängigen Künstler-Aktivisten, die im Niemandsland zwischen populären kulturellen Formen wie Spielen, verschiedenen Formen des Netzaktivismus (einschließlich Cyber-Feminismus) und theoretischen Zugängen operieren. Appropriation, Pastiche, Bricolage und andere postmoderne Tricks zählen zu ihren bevorzugten Ausdrucksmitteln.
Will die digitale Kunst in der Medienkultur des 21. Jahrhunderts einen Unterschied bewirken, muss sie ihren Schleier der Unschuld ablegen. Sie muss sich den problematischen, widersprüchlichen Realitäten der Cyberkultur stellen. Dabei muss sie wohl oder übel ihre eigenen internen Prozesse ebenso wie ihre Einstellung zu den herrschenden Systemen der Macht, Kontrolle und Kommerzialisierung, die sie zwangsläufig vereinnahmen, infiltrieren, dominieren und ihr öffentliches Image beeinflussen, kritisch hinterfragen und publik machen. So wie Wissenschaft und Technik nie völlig von den von Wirtschaft, Politik und Kultur ausgehenden Zwängen befreit werden können, kann die digitale Kunst nicht völlig „rein" und „frei" agieren, selbst wenn sie sich auf das Streben nach formaler, mathematischer, algorithmischer Schönheit beschränkt. Die Erforschung der Ästhetik digitaler Grammatiken bzw. der Funktionsweise von Code sind wichtige Ziele; die Erkenntnisse dieser Untersuchungen aus der Isolation des Maschinenraums zu befreien und im Bewusstsein der Cyber-Bürger – auch der Cyber-Art-Liebhaber – zu verankern, ist jedoch ein anderes, weitaus schwierigeres Unterfangen.

---

1 Negroponte, Nicholas: *The Architecture Machine,* MIT, Cambridge, Mass. 1970, S. 9.
2 Vgl. Matthew Fullers messerscharfe Analyse von Microsoft Word, „It looks like you're writing a letter: Microsoft Word", *http://www.axia.demon.co.uk/wordtext.html.*
3 Berger, John et al.: *Ways of Seeing,* BBC and Penguin Books, London and Harmondsworth, Middlesex 1972, S. 10.
4 „Schnittstellen" für den Diskurs über „Softwarekunst" sind der seit dem Jahr 2001 von der Transmediale in Berlin organisierte *software art award,* die Website *http://www.runme.org* und die *www.readme.org*-Aktivitäten. Die in diesen und anderen Foren tätigen Aktivisten stammen aus verschiedenen Ländern, hauptsächlich aus Europa.
5 Vgl. *http://www.walkerart.org/gallery9/lifesharing/.* Während der Biennale 2001 in Venedig zogen 0100101110101101.ORG, die eingeladen worden waren, ihre Werke im slowenischen Pavillon zu präsentieren, große Aufmerksamkeit auf sich, als sie die Entwicklung und Verbreitung eines Festival-Computervirus verkündeten. Mit diesem konzeptionellen Akt bestätigte die Gruppe gewissermaßen die Sicht der Softwarekunst-Jury der Transmediale 01, wonach „Computerviren als kritische Manifestation von Softwarekunst betrachtet werden können […]" (Transmediale 01, Erklärung der Jury, *http://www.transmediale.de/01/en/s_juryStatement.htm).*
6 Goodman, Cynthia: *Digital Visions. Computers and Art,* Harry N. Abrams & Everson Museum of Art, New York and Syracuse 1987, S. 24.
7 Reichardt, Jasia: *The Computer in Art,* Studio Vista, London 1971, vgl. S. 81 für eine Diskussion der *Computer Technique Group* und deren Theorie, dass „das Programm selbst das Kunstwerk ist".
8 Vgl. Franke, Herbert W.: *Computer Graphics Computer Art,* Phaidon, New York 1971, S. 107-108. Ein weiterer einflussreicher Theoretiker war Abraham Moles.
9 Benthall, Jonathan: „Science and Technology", in *Art Today,* S. 52, Praeger Publishers, New York 1972.

10 Es ist bekannt, dass Mignonneau für die technische Programmierung verantwortlich zeichnet, die zahlreichen Installationen dieser Gruppe sind jedoch Ergebnis einer sehr engen Zusammenarbeit aller Künstler, was die Darstellung des kreativen Inputs des Einzelnen unmöglich macht.
11 Rokebys *Very Nervous System* ist unter dem Namen *SVNS* auch als kommerzielles Softwarepaket erhältlich. Das Programm wird auch von zahlreichen anderen Künstlern verwendet.
12 Vgl. McCorduck, Pamela: *Aaron's Code,* W.H. Freeman, New York 1990.
13 In einer E-Mail erklärte Cohen vor kurzem: „Und wie bei jedem anderen seriösen Künstler, wird aufgegeben, was nicht weiterentwickelt wird. Die Antwort auf eine Ihrer Fragen ist daher, dass ich alte Programmversionen nicht lange aufbewahre. Ich könnte sie auch nicht verwenden, selbst wenn ich sie aufheben würde; sie wurden für andere Maschinen geschrieben und auf Medien gespeichert, die nicht mehr benutzt werden. Es wäre wahrscheinlich einfacher, frühere Versionen aus dem Gedächtnis heraus zu rekonstruieren, als die alten Medien wiederzubeleben: Ich würde es aber auch gar nicht versuchen." (Private E-Mail, 6. Mai 2003.)
14 Vgl. *http://www.kurzweilcyberart.com/.* Die Website enthält auch Informationen über die Geschichte von AARON, einschließlich eines Filmclips, in dem eine Zeichenmaschine in Aktion gezeigt wird. Man könnte auch behaupten, dass die Umwandlung von AARON in eine Desktopanwendung einer Trivialisierung des Programms gleichkommt. Dieses Gefühl hat vielleicht mit der enormen Rechnerleistung moderner Computer zu tun. AARON scheint die Bilder zu schnell, zu problemlos zu generieren. Dies ist natürlich nur ein subjektiver Eindruck, der mit der Komplexität des Codes nichts zu tun hat.
15 In Diskussionen verweist Cohen oft auf die Komplexität des Programms, das über drei Jahrzehnte hinweg kontinuierlich weiterentwickelt wurde. Obwohl es als Freeware angeboten wird, betrachtet Cohen AARON offensichtlich weiterhin als seine persönliche künstlerische Schöpfung. Die Freigabe des Quellcodes würde diese Funktion gefährden, allerdings auch zu einem radikaleren kollektiven Schöpfungsprozess führen.
16 *http://www.transmediale.de/01/en/s_juryStatement.htm.*
17 Cramer, Florian und Gabriel, Ulrike: „Software Art", *http://userpage.fu-berlin.de/~cantsin/homepage/writings/software_art/transmediale//software_art_-_transmediale.html.*
18 Eddo Stern hat aufgezeigt, wie Artefakte, Programmfehler und Netzwerkdefekte in interaktiven Multiplayer-Spielen dem Spieler zuweilen die digitale Architektur des Spiels vor Augen führen. Vgl. Sterns Artikel in *Mariosophy: The Culture of Electronic Games,* Hrsg. Huhtamo, Erkki und Kangas, Sonja, The University Press of Finland, Helsinki 2002 (auf Finnisch).
19 Vgl. Schleiner, Anne-Marie: „Velvet-Strike: War Times and Reality Games", *http://www.noemalab.com/sections/ideas/ideas_articles/schleiner_velvet_strike.html.* Schleiner hat auch wichtige Game-Patch-Kunstprojekte kuratiert.
20 Im Fall von *World Skin* geschieht dies, indem absichtlich künstliche, eingefrorene 2D-Bilder in einem navigierbaren 3D-Raum verwendet werden; im Fall von *That Sinking Feeling* durch animatronische Marionettenköpfe, die absichtlich die Äußerungen der Besucher missverstehen und Anzeichen eines schizophrenen Verhaltens aufweisen, das allerdings auch auf einen technischen Defekt zurückgeführt werden könnte.
21 Vgl. Shanken, Edward A.: *The House That Jack Built: Jack Burnham's Concept of 'Software' as a Metaphor for Art, http://www.duke.edu/~giftwrap/House.html.*
22 Burnham, Jack: „Notes on art and information processing", in *Software. Information technology: its new meaning for art* (catalogue), The Jewish Museum, New York 1970, S. 12.
23 SEEK erlangte traurige Berühmtheit, weil einige der Wüstenspringmäuse während der Ausstellung starben. Die Springmäuse fungierten als Symbol für den Menschen in einem technisch übersättigten Umfeld.
24 Vgl. Burnham, Jack: *The Structure of Art,* George Braziller, New York 1971.
25 Gidal, Peter, Hrsg.: *Structural Film Anthology,* BFI, London 1976, S. 14.
26 Gidal, Peter: *Materialist Film,* Routledge, London 1989, S. 17.
27 Sehr hilfreich für die weitere Diskussion dieses Aspekts ist: Le Grice, Malcolm: *Experimental Cinema in the Digital Age,* British Film Institute, London 2001. Le Grice war in der strukturellen Filmbewegung aktiv.
28 Vgl. Cox, Geoff, McLean, Alex and Ward, Adrian: „The Aesthetics of Generative Code", http://generative.net/papers/aesthetics/.
29 Cramer, Florian: „Concepts, Notations, Sofware, Art", *http://userpage.fu-berlin.de/~cantsin/homepage/writings/software_art/concept_notations//concepts_notations_software_art.html.*