netzspannung.org
Wissensraum für digitale Kunst und Kultur

TROGEMANN, Georg

Müssen Medienkünstler programmieren können?

Publiziert auf netzspannung.org:
http://netzspannung.org/positions/digital-transformations
02. Dezember 2004

Erstveröffentlichung: FLEISCHMANN, Monika; REINHARD, Ulrike (Hrsg.): Digitale Transformationen. Medienkunst als Schnittstelle von Kunst, Wissenschaft, Wirtschaft und Gesellschaft. Heidelberg: whois verlags- und vertriebsgesellschaft, 2004.

IMK
Fraunhofer Institut Medienkommunikation

The Exploratory Media Lab
MARS Media Arts & Research Studies

whois

GEORG TROGEMANN

MÜSSEN MEDIENKÜNSTLER PROGRAMMIEREN KÖNNEN?

Auf dem Holzschnitt mit dem Titel »Nova Scientia« aus dem Jahre 1537 stellt Nicolo Tartaglia (1499-1557) die damals herrschende Auffassung vom Verhältnis von Mathematik und Philosophie dar.

ABBILDUNG 1:
Nicolo Tartaglia, »Nova Scientia«, Holzschnitt (1537)

Am Eingang begrüßt Euklid die Studenten und geleitet sie in den ersten großen Zirkel, in dem sich die mathematischen Disziplinen Arithmetik, Musik, Geometrie und Astronomie befinden. Im linken Teil dieses Kreises wird mit Hilfe von Kanonenkugel auch die neue Vorstellung über Bewegungsbahnen gezeigt, die der lange gültigen Auffassung Aristoteles entgegensteht, dass bewegte Gegenstände senkrecht zu Boden fallen, sobald sie ihren »Impetus« verlieren. Dieses Wissen wird erst Jahrzehnte später durch die experimentellen Untersuchungen Galilei's bewiesen werden. Dahinter ist ein zweites Tor zu sehen, das in den kleineren Zirkel der Philosophie führt. Dort heißen Aristoteles und Platon die Schüler willkommen. Platon hält eine Rolle mit dem Wahlspruch seiner Akademie in der Hand. »Laßt niemanden hier eintreten, der der Geometrie unkundig ist«. Mit unserer Frage »Müssen Medienkünstler programmieren können?«, greifen wir dieses Bild in seiner modernen Fassung wieder auf. Aber ist die Frage nach Voraussetzungen, gar nach Fundamenten nicht längst überholt, unzeitgemäß und der postmodernen Kunst unangemessen?

Programmierung als Geistesgeschichte und Reflexionsform

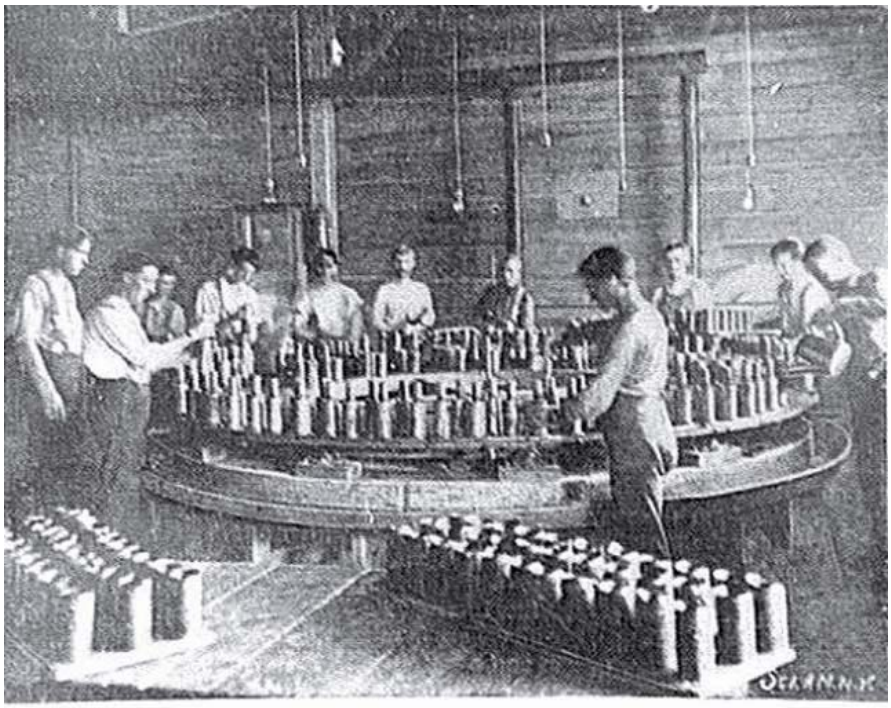
Zunächst gilt zu klären, was wir in unserem Zusammenhang unter Programmieren verstehen wollen, und warum diese spezielle Tätigkeit überhaupt von allgemeinem Interesse sein könnte. Hierzu ist allerdings weiter auszuholen. Schon in der Antike war bekannt, dass wir, um die Welt zu gestalten und Grund legend zu verändern, über keine explizite Theorie verfügen müssen. Was die Griechen unter »•••••« verstanden, waren primär diejenigen Fertigkeiten und Geschicklichkeiten, die bestimmte Leistungen und Produkte hervorzubringen vermochten und die man durch Abschauen und Nachmachen erlernen konnte.[1] Technik heißt dort also, dass man sich auf eine Sache versteht, ohne die Sache selbst zu verstehen. Die tüchtigen Handwerker, die Sokrates mit seiner strapaziösen Fragerei quälte, waren natürlich nicht in der Lage ihm zu erklären, was sie da eigentlich tun, wenn sie ihren alltäglichen Beschäftigungen nachgehen. Diese Trennung von Sachverstand und Sachbeherrschung hat den Siegeszug der Industrialisierung ermöglicht und ist bis heute ein wesentliches Kennzeichen unseres Verhältnisses zur Technik. Das bloße Sich-Verstehen-auf unterscheidet sich Grund legend vom wissenschaftlich oder künstlerisch reflektierten Einlassen auf die Erscheinungen. Wenn wir hier von Programmierung sprechen, dann ist neben der reinen Fertigkeit des Codierens immer auch die Geistesgeschichte der programmierbaren Maschine und der reflektierte Umgang mit dem Eigenwesen dieses Artefaktes gemeint. Die übliche strikte Trennung zwischen praktischer Einlassung und kritischer medien- oder kunsttheoretischer Reflexion, ist zu bekämpfen. Arno Bammé fordert von einen integrierten Programmieransatz[2]:

»Programmieren ist hier also nicht Selbstzweck, sondern ein Mittel der Erkenntnis, um die Logik des Computers zu erfahren und das eigene Verhältnis hierzu sich bewusst zu machen. Der wesentliche Maschinenbestandteil des Computers ist sein Programm; programmieren bedeutet, Maschinen zu konstruieren.«

Nach Lewis Mumford geht die Idee der Maschine bereits auf die Hochkultur des Gottkönigtums in den frühen Perioden des Pyramidenzeitalters zurück. Die erste Maschine – die Mumford Megamaschine nennt – besteht in ihren einzelnen Komponenten allerdings nicht aus mechanischen Elementen, sondern aus Menschen, deshalb war sie lange unsichtbar und wurde nicht als Vorläufer unserer heutigen Maschinen erkannt. Trotzdem hat sie als Arbeitsmaschine über eine enorme Produktionskraft verfügt, denn ihre Arbeitsleistung dokumentiert sie bis heute mit ihren Bauwerken, unter anderem der großen Pyramide von Gizeh.[3] In diesem Sinne steht Programmierung in einer mindestens 5000 jährigen Tradition und spiegelt archetypische Grundfiguren des menschlichen Denkens wider. Die Struktur der Maschine, wie wir sie im frühzeitlichen Ägypten finden, unterscheidet sich auf dieser Ebene wenig von der Manufaktur des Industriezeitalters oder der »von Neumann«-Maschine des so genannten Informationszeitalters, es handelt sich um universelle Prinzipien.

»Technologie ist nichts anderes als die gegenständlich gewordene Widerspiegelung der menschlichen Seele in die Natur. Maschinen sind vom Menschen produziert. Sie sind nichts anderes als die Materialisierung dessen, was im Kopf, in der Psyche des Menschen bereits vorhanden ist. Maschinen können als materialisierte Projektionen von Wesensmerkmalen des Menschen begriffen werden. Nicht die Technik wäre dann das größte Problem des gegenwärtigen Menschen, sondern der Mensch selbst.«[4]

Es fehlt hier der Platz, um all dies eingehender zu betrachten und die Stationen der Programmierung vom Pyramidenbau bis zur Turingschen Universalmaschine nachzuzeichnen.[5]



Casting Blank Records.

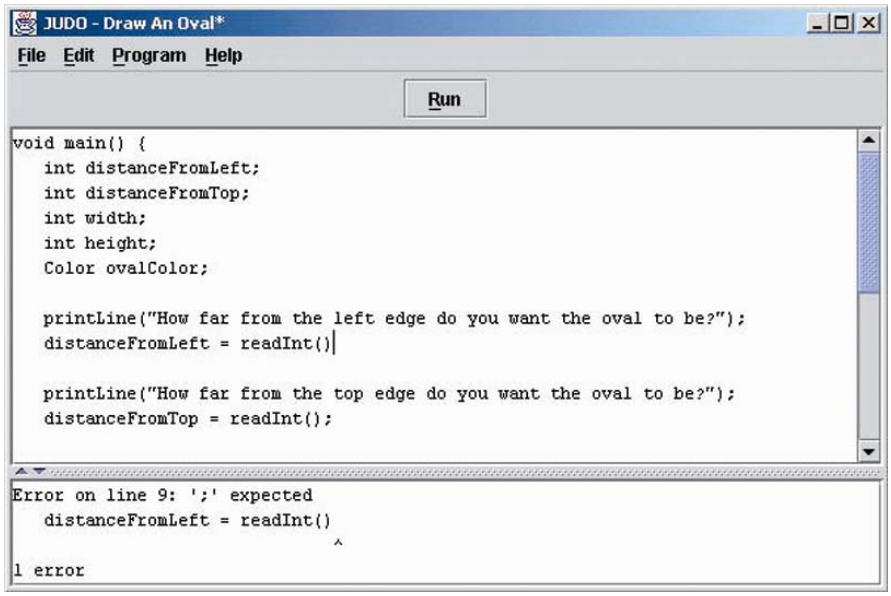


ABBILDUNG 2 und 3:
Manufaktur und Programmcode als gegenständlich gewordene Widerspiegelungen archetypischer Denkformen und universeller Prinzipien.

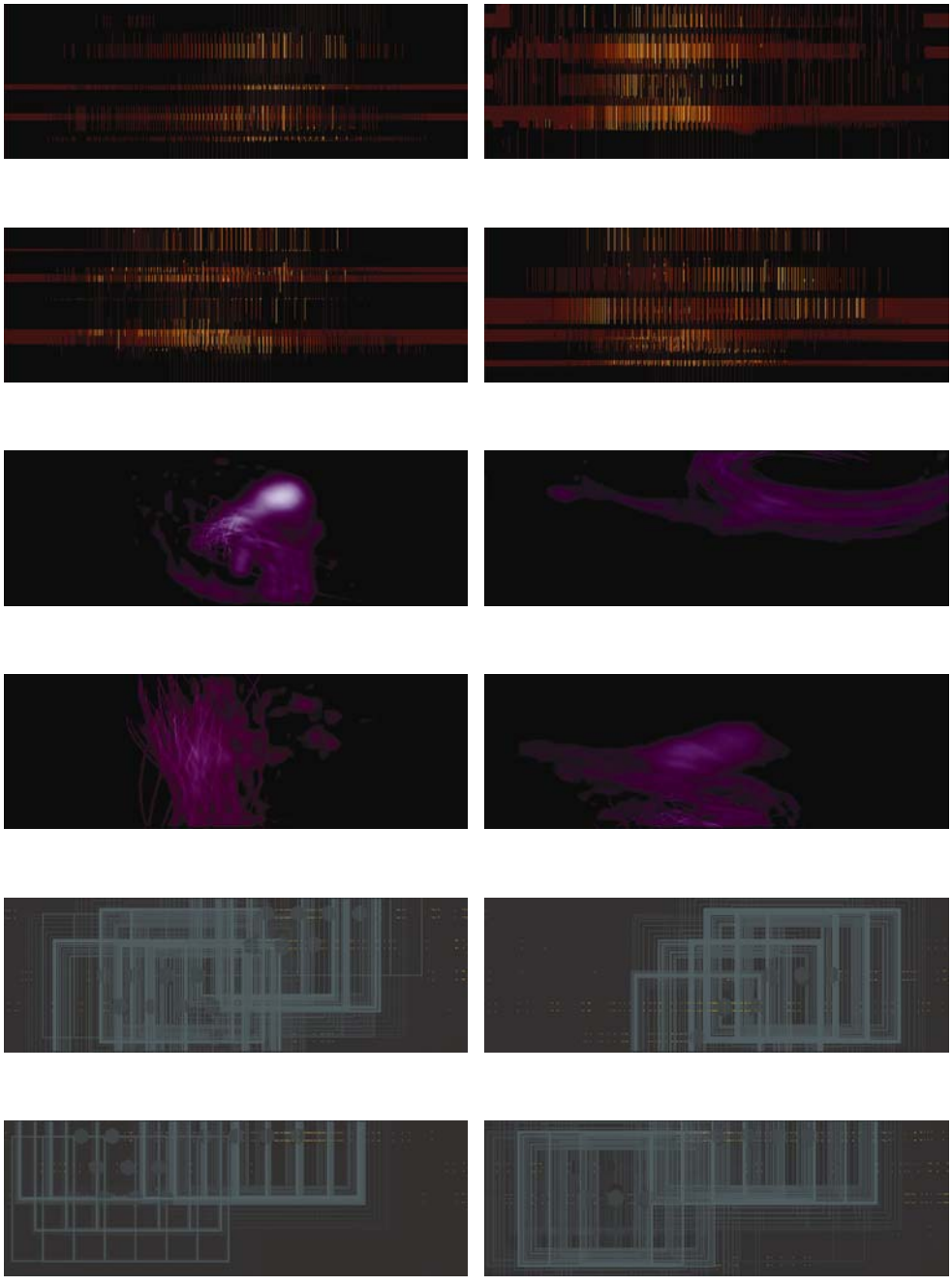
Über den Eigensinn[6] der Medien

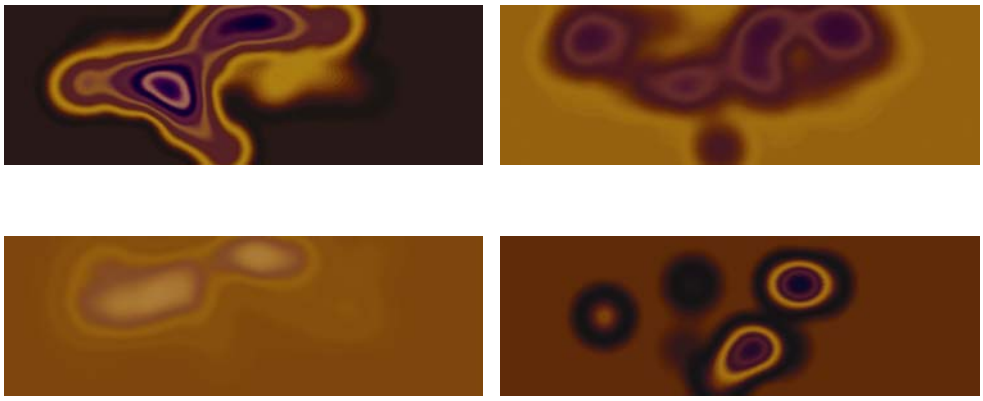
»... perhaps information technology has become [as Jack Burnham in his theory demanded] pervasively, if not subconsciously present in the lifestyle of our culture, such that its aesthetic implications are sufficiently manifest to play a constructive role in proposing new artistic paradigms.«[7]

Dieses Zitat fasst unsere gegenwärtige Situation und den potentiellen Einfluss des Computers auf die Kunst pointiert zusammen. (Über die Umkehrung, den Einfluss der Kunst auf die Entwicklung der neuen Technologien, gibt es unserer Ansicht nach derzeit weniger zu berichten.) Mit der Omnipräsenz der Informationstechnologien und dem Erscheinen der Interfaces als Vermittlungsinstanz zwischen den logischen Strukturen der Maschine und den Wahrnehmungsprozessen ihrer Benutzer wurde das Spiel zwischen Mensch und Maschine zunächst allerdings noch komplizierter, als es sowieso schon immer war. Die Relationen und Spannungen zwischen Imagination, Sprache und Interface wirken gerade am Punkt der Programmierung in besonderer Weise zusammen. Programme sind nicht länger nur textlich verfasste Repräsentationen stringenter mathematischer und naturwissenschaftlicher Modelle, so wie sie es bis in die 80er Jahre des 20. Jahrhunderts meist waren, sondern sie dringen unaufhaltsam in den unscharfen Bereich der gesellschaftlich und sozial codierten Nachrichten und Wahrnehmungsformen vor. Die bisherige Sichtweise der Maschine als rein logisch-technische Reflexionsform muss hier deshalb eine Ausweitung erfahren. Imaginationen – also Vorstellungen, Phantasien, Träume, Utopien – werden zunächst in Symbole, das heißt sprachliche Modelle der Maschine transformiert, um dort prozessiert zu werden und als wahrnehmbare Ereignisse wieder an die Oberfläche der Maschine zurückzukehren. Das Dreiecksverhältnis zwischen Imagination, Sprache und Material (Interface) – das gleichzeitig ein Grundmuster jedes kulturellen Prozesses ist – tritt auf der Ebene der Programmierung besonders deutlich in Erscheinung und kann hier nun experimentell untersucht werden.

Der Begriff des Eigensinns der Medien bei Giaco Schiesser geht davon aus, dass alle Medien etwas je eigenes auszeichnet, dass also auch Computer und Netzwerke spezifische Potenziale, Strukturen und Beschränkungen zeigen. Insbesondere die Geschichte der Kunst des 20. Jahrhunderts macht deutlich, dass jedes Material Ausgangsbasis für die künstlerische Praxis und die künstlerische Auseinandersetzung sein kann. Gleichzeitig gilt aber, dass, nachdem einmal die Festlegung auf bestimmtes Material erfolgt ist, man sich dann auf den Eigensinn dieses Mediums einlassen muss. Diesen Eigensinn gilt es für die Computertechnologien noch herauszuarbeiten, wir stehen dabei am Anfang. Ich weiß allerdings nicht, wie die neuen Experimentierräume, »... in denen neugierig, radikal und kompromisslos die individuelle und kollektive, eigensinnige Arbeit der StudentInnen an selbst gewählten oder auferlegten, Interessen/Inhalten/Themen und ihre Arbeit am und mit dem Eigensinn unterschiedlicher Einzel- und Hybrid-Medien gefordert und gefördert wird«[6], sonst aussehen sollen, wenn dort nicht ganz selbstverständlich die Sprache der Neuen Medien, das heißt die Programmiersprache, gesprochen wird. Natürlich stellt das Erlernen dieser Sprache zunächst eine Hürde dar, sobald wir diese aber genommen haben, steht uns das Medium in seiner ganzen Multiperspektivität offen – die weit über den bloßen Werkzeuggebrauch hinausreicht und sich auch nicht in den ewigen Variationen bekannter Wahrnehmungsformen erschöpft. Erst innerhalb des neuen Horizontes können wir alle künstlerischen Strategien und Herangehensweisen konsequent durchspielen.
Am weitesten ist die konsequente Nutzung von Programmcode als künstlerisches Material derzeit sicherlich im Bereich der Computermusik entwickelt. Hier gibt es eine lange Tradition, die mit Iannis Xenakis auch bereits einen Künstler vorzuweisen hat, der Codes und komplexe formale, mathematische Methoden ins Zentrum seiner Musiktheorie stellte und es damit zu weltweiter künstlerischer Anerkennung gebracht hat. Aber auch für die junge Sound-Generation ist elektroakustische Musik nicht denkbar, ohne Kenntnisse in Signalverarbeitung und den damit einhergehenden formalen Methoden. Mit Max/MSP und Pure Data stehen Werkzeuge zur Verfügung, die von Künstlern für ihre eigenen Belange entworfen wurden; mit Super Collider wird sogar die erste Programmiersprache von kreativen Musikern mit- und weiterentwickelt und explizit auf ihre Anforderungen der Soundgenerierung und -kontrolle zugeschnitten. Aber auch algorithmische

Untersuchungen auf visueller Ebene können auf eine lange Tradition zurückblicken. Mit Roman Verostko gibt es hier zum Beispiel einen künstlerischen Veteranen, der seit über 40 Jahren mit Algorithmen experimentiert und Software als Genotyp seiner epigenetischen Bilder begreift. In Form der »Generativen Informationsästhetik« – die wesentlich geprägt war durch Max Bense und seine Schüler – existierte bereits in den 60er und 70er Jahren in Deutschland ein theoretisch fundierter Ansatz, der Algorithmik und Ästhetik zu verbinden suchte. Ansätze, die solches auf zeitgemäßer technologischer wie theoretischer Basis und Kenntnis versuchen, müssen wir leider vermissen. Mit dem Projekt »Processing« von Ben Fry und Casey Reas existiert zumindest ein neuer engagierter, wenn auch in erster Linie Praxis orientierter Versuch, junge »hybride« Künstler an die Programmierung heranzuführen.





ABBILDUNGEN 4:
DAKADAKA, 2000-2004, Casey Reas und Golan Levin

»Typing is a percussive spatial action — a play of tiny thoughts scattered onto a tightly organized grid. Typing is also a kind of speech, spoken through the fingers with flashing rhythms and continuous gestures. Dakadaka is interactive software that explores these two ideas by combining positional typographic systems with an abstract dynamic display. Dakadaka was created in collaboration with Golan Levin.«

Für die relativ neuen Bereiche komplexer Interfaces gibt es inzwischen ebenfalls zahlreiche Arbeiten, die als Beispiele für avancierte künstlerische Experimente dienen können. Diese hybriden Künstler begreifen den Programmcode als wesentliche Komponente ihrer Projekte und können sich nicht vorstellen, diesen Bereich ohne Authentizitätsverlust an andere zu delegieren. Eine fundierte Theorie, die Programmstrukturen als universelle Prinzipien und archetypische Denkformen begreift und diese mit zeitgemäßen Kunst- und Ästhetiktheorien verbindet, steht allerdings noch aus.

Zusammenfassung

Medienkunst darf – wenn sie sich behaupten will – nicht nur als Sammelbegriff fungieren für alle möglichen künstlerischen Arbeiten die am oder auch nur in der Nähe des Computers entstehen. Zugleich darf Medienkunst nicht ausschließlich der Wahrnehmung verpflichtet bleiben, nicht nur, weil sie sonst nur allzu leicht dem alten Ideal des Gesamtkunstwerkes verfallen wird – wie es derzeit im Gewand der virtuellen Realität reanimiert wird – sondern auch, weil sie dann dem Eigensinn des Mediums nicht gerecht wird. Die Ebene der Programmcodes zielt auf eine ganz andere Form der Generalisierung und der Auseinandersetzung mit der Maschine ab. Codes abstrahieren zunächst von den konkreten Medienausprägungen wie Bild, Text oder Ton und haben ihren Ursprung in menschlichen Denkprozessen und ihrer Reflexion. Dies ist aber tatsächlich nur die eine Ebene des Codes. Da die neuen Interfaces – vermittelt durch Codes – zunehmend kulturell und sozial codierte Wahrnehmungen des Benutzers ansprechen, müssen in der Programmierung nun nachhaltig Logik und Wahrnehmung miteinander verbunden und geeignet ins Visuelle und Akustische übersetzt werden. Den traditionellen Schulen der Wahrnehmung mit allen Sinnen muss deshalb beim Umgang mit Neuen Medien eine Schule der Kognition an die Seite gestellt werden.

Die Ausgangsfrage unseres Beitrags ist abschließend radikaler zu formulieren. Es reicht wahrscheinlich nicht aus, zu fordern, dass lediglich Medienautoren und insbesondere Medienkünstler sich mit der Struktur der programmierbaren Maschine auseinandersetzen. Vielmehr stellt sich die Frage, wie sich Künstler heute überhaupt noch an eine Leinwand stellen, oder einen Steinblock bearbeiten können, ohne die Kulturgeschichte der programmierbaren Maschine – mit ihren alles

durchdringenden Einflüssen auf die gegenwärtige Gesellschaft und unsere Existenz – als bewussten Hintergrund ihrer eigenen Arbeit und ihres eigenen Lebens zu begreifen. Die Vorstellung, dass Kunst von Alltagseinflüssen getrennt werden kann, ist eine ähnliche Idealisierung wie die Schimäre der »Objektivität« in den exakten Wissenschaften. Oswald Wiener drückt unsere Forderung im Vorwort seiner Einführung in die Theorie und Programmierung der Turing-Maschinen folgendermaßen aus[8]:

»In einer Welt von Computer-Benutzern, die – nicht zufällig – zugleich eine Welt der mechanistischen Erklärungen ist, sollte ein beträchtlicher Teil des in diesem Buch Gebotenen zur Allgemeinbildung gehören. ... Wie immer man sich das Verhältnis von Erkenntnis zu Erkanntem wünschen mag: Man sollte recht genau wissen, was eine mechanistische Erklärung ist, wenn man für oder gegen sie zu Felde zieht. Mit der Allgemeinbildung steht es indessen in all diesen Hinsichten nicht zum Besten.«

---

1 BLUMENBERG, Hans: Wirklichkeiten in denen wir leben. Stuttgart: Reclam, 1999, S. 12.

2 BAMMÉ, Arno et. al.: Technologische Zivilisation und die Transformation des Wissens. Profil Verlag, 1988, S. 42.

3 MUMFORD, Lewis: Mythos der Maschine: Kultur, Technik und Macht. Frankfurt am Main: Fischer Taschenbuch Verlag, 1977, S. 219 ff.

4 BAMMÉ, Arno et. al: Maschinen-Menschen Mensch-Maschinen: Grundrisse einer sozialen Beziehung. Hamburg: Rowohlt Taschenbuch, 1983, S. 110.

5 TROGEMANN, Georg; VIEHOFF, Jochen: Code Art – Eine elementare Einführung in die Programmierung als künstlerische Praktik. Wien: Springer-Verlag, erscheint Ende 2004.

6 SCHIESSER, Giaco: Media | Art | Education: Arbeit am und mit dem Eigensinn. In: CODE - The Language of our Time, Ars Electronica 2003, Ostfildern: Hatje Cantz Verlag, S. 371 – 373.

7 SHANKEN, Edward A.: The House That Jack Built: Jack Burnham's Concept of »Software« as a Metaphor for Art. In: Leonardo Electronic Almanac 6:10, November 1998.

8 WIENER, Oswald: Eine elementare Einführung in die Theorie der Turing-Maschinen. Wien: Springer-Verlag, 1998.