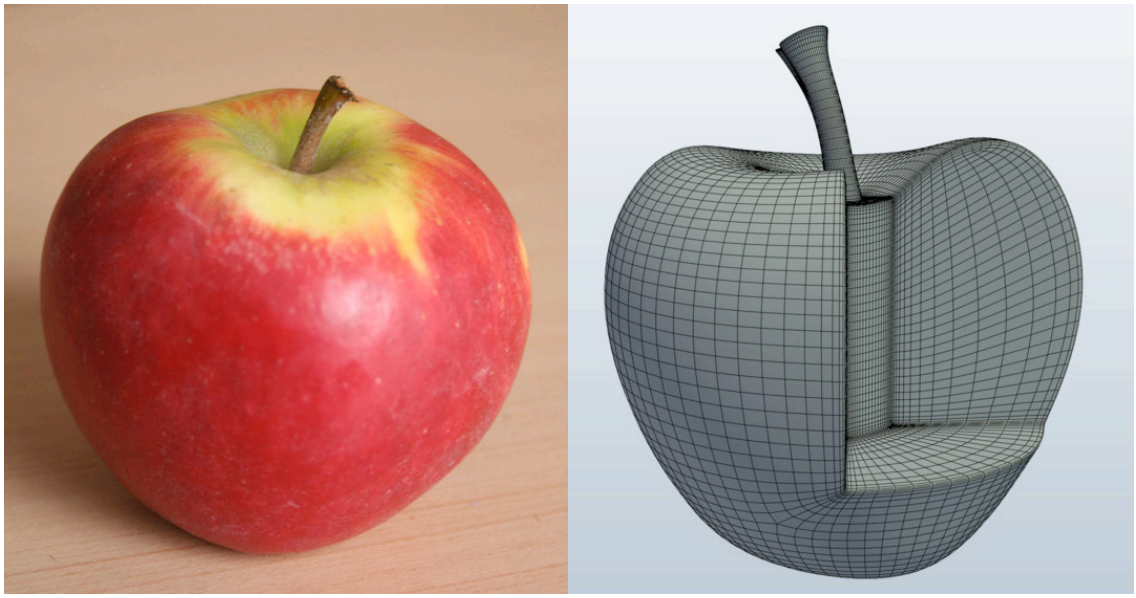
Die Fülle des Konkreten am Skelett des Formalen

Über Abstraktion und Konkretisierung im algorithmischen Denken und Tun



Prolegomenon

Können wir heute noch einen Apfel anschauen, ohne an Vitamine, Mineralien und Ballaststoffe zu denken? Hat die Abstraktion unser Denken so sehr vereinnahmt, dass wir eine enge Verwandtschaft zwischen einer Frucht und einer Vitamintablette annehmen? Durch die Quantifizierung und Reduktion auf messbare Größen wird alles komparabel, auch Ungleiches. Und sobald wir einen Sachverhalt erst einmal auf die Zahl gebracht haben, folgt unweigerlich seine weitere Verrechnung in allerlei Modellen. Die Eliminierung von Qualitäten und die Funktionalisierung von Erfahrung hat längst von der Wissenschaft auf den Alltag übergegriffen. Im Zentrum dieses Prozesses steht der Computer. Er ist das Werkzeug einer formalen Rationalität, mit der wir unsere Naturbeherrschung auf allen Ebenen vorantreiben. Die ehemals aufs Ganze zielende Vernunft scheint in gleichem Maße abzunehmen wie die abstrakte instrumentelle Rationalität zunimmt. Wir formen unser Leben nach den gleichen Gesetzen, nach denen wir auch unsere technischen Apparate konstruieren. Zwar wächst das Volumen der bekannten Fakten dabei rasant. Die Gefahr ist aber, dass wir uns den Eigengesetzlichkeiten und inneren Logiken des Messens, Zählens und Funktionalisierens ausliefern.

Dabei markieren die Quantifizierung, Formalisierung und Funktionalisierung der Lebenswelt nur den Weg des Konkreten in die Maschine. Im Prozess des Rechnens kommen die Phänomene und Kontingenzen zurück. Diese Rückbindung an die Welt wollen wir im Folgenden betrachten.

Die Fülle des Konkreten am Skelett des Formalen

Über Abstraktion und Konkretisierung im algorithmischen Denken und Tun

© GEORG TROGEMANN (2014)
Kunsthochschule für Medien Köln
interface.khm.de

Abstract

Softwaresysteme sind eine effektive technische Strategie zur *Herstellung erfundener Wirklichkeiten*. In unserer digitalen Kultur handeln programmierte Systeme bereits in beträchtlichem Maße anstelle von Menschen. Dabei gelingt wechselseitiges Aufeinander-Reagieren von Menschen und programmgesteuerten Apparaturen nur dort, wo Software und Anwender sich auf einen gemeinsamen, oft unsichtbaren Kontext beziehen.

Noch weiß die Apparatur nichts von ihrem Tun, es werden lediglich externe Ereignisse registriert und diese mit inneren Zuständen zu Antworten und Reaktionen verrechnet. Die elektronische Hardware fungiert dabei als Substrat und offenes Gefäß, das unterschiedliche Programme aufnehmen und so unterschiedliche Logiken der Interaktion realisieren kann. Im Programm selbst finden wir nur ein Spiel von Zeichen, das Größen und Symbole nach festen Regeln in Beziehung setzt und das Ergebnis in den Displays anzeigt oder mit Hilfe von Aktoren in Handlungen übersetzt. Wie aber halten die Zeichen die Verbindung zur Welt? Oder andersherum: Wie kommt ein Stück Welt in die Maschine? Am Beispiel der Geometrie wird in groben Zügen nachvollzogen, welche Abstraktionen, Idealisierungen und Modellvorstellungen im Spiel sind, bis geometrische Primitive als manipulierbare Objekte im Computer zur Verfügung stehen.

Die verschiedenen Abstraktionsebenen, vom Anwendungsmodell über verschiedene formale Spezifikations- und Softwareebenen bis zum binären Prozessorbefehl sind ein gut untersuchtes Feld im Software-Engineering. Doch die Quantifizierung und Formalisierung markiert nur den Weg in die Maschine und damit nur die Hälfte der Strecke. Im Vollzug des Rechnens findet eine Konkretisierung und Rekontextualisierung des Formalen statt. Zeichen werden rücktransformiert in Kontingenz und wahrnehmbare Qualität. Während im Zuge der Abstraktion eine radikale Säuberung des modellierten Weltausschnitts erfolgt die alles Sinnliche entfernt, sehen wir beim Prozessieren der Algorithmen die Umkehrung. Das nackte Skelett wird wieder angereichert und die ganze Fülle an Gedanken, Gefühlen und Interpretationsmöglichkeiten entfaltet sich aufs Neue. Doch wird nicht zurückgepackt was ehemals weggenommen wurde, sondern Anderes, das sich aus unterschiedlichen Quellen speist. Hier, in den Leerstellen der formalen Beschreibungen, nisten wesentliche Anteile des Reichtums, der Vielfalt und Qualität des Digitalen. Das Formale und das Konkrete dürfen aber nicht als Widersacher im Ringen um Fülle und Ästhetik betrachtet werden, sie bilden ein kraftvolles Gespann.

Kluger Umgang könnte darin bestehen, die prinzipiellen Grenzen und Möglichkeiten des Formalen zu kennen, sich der *instrumentellen Vernunft*[1] aber nicht auszuliefern, sondern sie in ihrer Wechselwirkung mit dem Konkreten zu untersuchen und fruchtbar zu machen.

[1] Von Max Horkheimer geprägter Begriff.

«*Also: Wäre es nicht die Aufgabe elektronischer Künste, besonders elektronischer Unterhaltungsmusik, gerade nicht diese Maschinenfrage zu 'radikalisieren', 'konzeptualisieren' und so fort [...], sondern endlich mal erfahrbar zu machen, wie unglaublich normal und nicht so schlimm und einfach total okay diese Sache mit den Maschinen ist, die wir da haben und benutzen können, eben als Alltag, wie er wäre, wenn er nicht unter der Bestimmung des allgemeinen Elends stünde [...], sondern unter der Bestimmung der Vernunft, die sagt: Ja, es gehört so*».[2] Dietmar Dath

DIE NORMALITÄT DES DIGITALEN

Nehmen wir an, «*diese Sache mit den Maschinen*» ist tatsächlich ganz normal und wir kommen mit der Plattitüde von der revolutionären Kraft der Computer nun endgültig nicht mehr durch. Akzeptieren wir also, dass die ehemals 'Neuen Medien' mittlerweile eine unhintergehbare Folie des Alltags bilden. Fragen der folgenden Art sollten dann leicht zu beantworten sein: Was genau machen wir eigentlich, wenn wir Algorithmen entwickeln, implementieren und auf Rechnern ausführen? Ist es nur ein Tun oder Verstehen wir auch? Ist es nicht vielmehr immer so, dass dort, wo es um Handlungs- und Verfügungswissen geht, wir mehr tun können, als wir verstehen? Wie kommt es, dass sich schon bei relativ kleinen Programmen die Auswirkungen, Verhaltensweisen und produzierten Ästhetiken nur durch umfangreiche Tests und iterative Prozesse einigermaßen zuverlässig vorhersehen lassen? Solche Fragen zielen nicht vorrangig auf die Rezeption medialer Inhalte ab, sondern auf die Bedingungen ihrer Herstellung. Wir wollen uns diesen Voraussetzungen von der Material- und der Formseite nähern.

Um Programmcodes als Material und als Form computerzentrierten Arbeitens aufgreifen und thematisieren zu können, müssen wir das Umfeld ansehen in dem sie entstehen und wirksam werden. Computeranwendungen sind heute lose, sich je neu formierende Konfigurationen aus Hardwarenetzen, Algorithmen und Interfaces [siehe Anhang A]. Eine besondere Herausforderung dieses komplizierten Zusammenspiels ergibt sich aus dem intrinsischen Spannungsverhältnis von Vernunft und Ästhetik. Im künstlerischen Umfeld gilt es dieses Zusammenwirken der technikrationalen und der ästhetischen Schichten und ihrer jeweiligen Traditionen und Eigenheiten zu untersuchen. Um die Ausprägungen der digitalen Kultur besser zu verstehen, müssen wir dafür einerseits zwischen Form und Inhalt vermitteln, andererseits müssen wir die Materialität der Maschine zurück in den Diskurs bringen. Es sind weder die algorithmischen Zeichen noch das Material, die jeweils alleine die digitalen Inhalte, also Klang-, Bild-, Verhaltens- oder Objekteigenschaften bestimmen könnten. Es ist die Verbindung aus beiden, etwas das man als 'informiertes Material' bezeichnen könnte. Form, Materialität und Inhalt spielen sich bei der Ausführung der Algorithmen und der Rückbettung in den Kontext der Anwendung gegenseitig in die Hände. Das durch das Konzept der programmierbaren Maschine künstlich getrennte muss für das Verstehen seiner Anwendungen wieder zusammengeführt werden.

Gleichzeitig gibt es in der digitalen Kultur fundamentale Verschiebungen im Herrschaftsverhältnis zwischen Form und Inhalt. «Die auf der zweiwertigen Logik beruhenden formalen Strukturen sind in ihrer gesellschaftlichen Bedeutung stets unterschätzt worden. Sie wurden als sekundär begriffen, eben nur als Form eines Inhalts. Erst wenn man die Form als etwas Eigenes, vom Inhalt Unabhängiges begreift, das eine eigene Realität darstellt, wird ihr Beitrag zur gesellschaftlichen Synthese deutlich.»[3] Im Hinblick auf die gesellschaftliche und kulturelle Relevanz übernimmt also die Form die Führung und der Inhalt wird nachgeordnet. Für das Verständnis der Bedingungen und Wirkungsweisen unserer digitalen Produkte sind Form und Inhalt aber als Einheit zu denken.

[2] Dietmar Dath, in: Klaus Sander, Jan St. Werner, *Vorgemischte Welt*, Suhrkamp Verlag 2005, S. 9.
[3] Eggert Holling, Peter Kempin, *Identität, Geist und Maschine*, Rowohlts Enzyklopädie 1989, S. 13.

Unsere Vernunft ist keine konstante Gegebenheit, sondern Ergebnis eines geschichtlichen Entwicklungsprozesses.[4] Wenn aber die Vernunft nie statisch war, sondern einer andauernden Entwicklung unterliegt, wie könnte dann unsere Vorstellung vom Menschen konstant sein? Alfred Sohn-Rethel hat für die Warenform herausgestellt, wie die menschlichen Denkformen – insbesondere die Fähigkeit zur Abstraktion – vom gesellschaftlichen Sein bestimmt werden. Den Ursprung des abstrakten Denkens macht er an der Tauschabstraktion fest, Warenform und Denkform sind für ihn zusammengehörige Elemente derselben Formation. Abstraktes logisches Denken und Geld in Münzform treten deshalb nicht zufällig gleichzeitig um 680 v. Chr. in Ionien auf. Heute fragen Anthropologen wie Paul Rabinow deshalb nicht mehr nach einem Menschsein als solchem, sondern nach der jeweiligen den Menschen konstituierenden Vernunft. Weder das Soziale, noch die Kultur werden als Konstante betrachtet. Vielmehr sind die Vernunftformen in Kultur und Gesellschaft auf ihr anthropologisches Potential hin zu untersuchen. «In der Geschichte der Gegenwart gälte es die 'Rede vom Menschen' auf ihre Vernunft- und Rationalitätsformen hin zu untersuchen und in ihrer Produktivität auf einen immer im Werden begriffenen Menschen hin zu analysieren, wobei ihre Aufgabe nicht zuletzt darin bestünde, die Grenzen der jeweiligen Vernunft zu bestimmen.»[5] Der Schlüssel zur Vernunft wiederum ist die Sprache. «Der Mensch ist ein 'vernünftiges' Wesen: in dem Sinne, dass die 'Vernunft' aus der Sprache stammt und unlöslich an sie gebunden ist – dass ratio und oratio, Sprechen und Denken, Wechselbegriffe werden.»[6]

Was ist aber gemeint, wenn wir sagen, dass die rationalen Strukturen, auf denen unsere digitale Kultur gründet, sich in unsere Vernunft einschreiben? Auf welche Weise hat sich beispielsweise die Warenform des geldbasierten Tausches in die Vernunftstrukturen der Gesellschaft eingeprägt? Es muss sich aus der Fülle der Phänomene, der Handlungsweisen und Wahrnehmungen etwas herauslösen lassen, das, sobald es begrifflich gefasst ist, eine Eigendynamik entwickelt die nur noch ihren inneren Gesetzmäßigkeiten folgt. Der *Impact Factor* zur Messung wissenschaftlicher Relevanz, die *evidenzbasierte Medizin*, die *Quantified Self Bewegung*, der *Kunstkompass* zum Ranking von Künstlern können als aktuelle Beispiele für die Zunahme quantifizierender Entscheidungssysteme dienen. Sobald solche Strukturen sich ausgebildet haben und installiert sind, verschieben sich tradierte Handlungsstrategien und Ziele. Im Extremfall wird nicht mehr versucht gute Wissenschaft zu machen, sondern einen möglichst hohen *Impact Factor* zu erhalten, nicht mehr das subjektive Befinden ist wichtig, sondern die Messwerte des *Self-Trackings*. Eine Form schreibt sich in die Vernunft ein, wenn umfassende Sichtweisen aufgegebenen und der inneren Gesetzlichkeit, d.h. der Form und Eigenlogik eines abgrenzbaren, künstlichen Orientierungsrahmens gefolgt wird.

Solche Prozesse sind nicht neu, aber erst die Omnipräsenz der Computer in allen Lebensbereichen ermöglicht in den meisten Fällen die Installation quantifizierender, algorithmischer Entscheidungssysteme, mindestens aber beschleunigt sie deren kulturelle Sedimentierung. Auswirkungen sehen wir Beispielsweise im Finanzsektor. Der Finanzmarkt handelt im Gegensatz zur Realwirtschaft ausschließlich mit Kapital. Nun ist die Geldform an sich schon eine dominante, die Gesellschaft stark prägende Formalstruktur, die gleichwohl einfach und – da sie Form ist – inhaltslos bleibt. Aber gerade deshalb eignet sich die Geldform besonders, um zu zeigen, wie Kapital und Algorithmus sich ergänzen und zusammen eine neue Eigenlogik entfalten. Der Hochfrequenzhandel ist eine Sonderform automatisierten oder algorithmischen Handels, der heute schon 40 Prozent des Handelsvolumens an europäischen und 70 Prozent an US-amerikanischen Börsen

[4] Alfred Sohn-Rethel, *Das Geld, die bare Münze des Apriori*, Wagenbach 1990, siehe dazu auch: Georg Trogemann, Jochen Viehoff, *CodeArt – Eine elementare Einführung in die Programmierung als künstlerische Praktik*, Springer-Verlag, Wien 2005

[5] Carlo Caduff, Tobias Rees, in: Paul Rabinow, Anthropologie der Vernunft – Studien zu Wissenschaft und Lebensführung, Suhrkamp 2004, S. 7.

[6] Ernst Cassirer, Form und Technik, in: Peter Fischer, *Technikphilosophie*, Reclam Verlag 1996.

ausmacht.[7] Computer bewegen hier in Sekundenbruchteilen riesige Finanzsummen. Durch blitzschnelle Analyse von Börsendaten werden kleinste Kursschwankungen ausgenutzt um Börsenorders zu platzieren. Die Kaufs- und Verkaufsentscheidungen treffen dabei konkurrierende Algorithmen. Es kommunizieren nur noch Maschinen mit Maschinen, der Mensch ist Zuschauer. Die hierfür entwickelten algorithmischen Strategien folgen der Eigenlogik eines formalisierten und enorm beschleunigten Basars. So genannte *Sniffer* versuchen andere Algorithmen aufzuspüren und zu überlisten oder es werden Täuschmanöver ausgeführt, bei denen Algorithmen mit sich selbst handeln, um Umsatz vorzutäuschen. Gleichzeitig findet ein Streit der Kulturen statt. Wirtschaftswissenschaftler versuchen Modelle auf der Basis ökonomischer Theorien zu entwickeln, Physiker, Informatiker und Mathematiker betrachten den Handel dagegen als abstrakte Struktur, für die sich jedes Modell eignet, solange eine innere Strukturähnlichkeit zum Börsenhandel gewährleistet bleibt. Als interessante Randnotiz kann erwähnt werden, dass selbst in diesem streng algorithmischen Spiel die Materialität des Rechnens nicht vollständig auszuschalten ist. Da für Kaufs- und Verkaufsentscheidungen beim so genannten *Algo-Trading* die Millisekunde zählt, spielt die physische Nähe zwischen den handelnden Rechnern eine entscheidende Rolle. Da die Verbindungsleitungen dann am kürzesten sind, wenn Börsenrechner und Handelsrechner direkt neben einander stehen, sind die Plätze in der unmittelbaren Nähe der Börsenrechner auch am begehrtesten und teuersten. In der Vergangenheit kam es beim Algo-Trading immer wieder zu unerklärlichen Kurseinbrüchen und sogar kurzzeitigen Totalzusammenbrüchen von Börsen. Solche Aussetzer bleiben dann nicht systemintern, sondern können auf die Realwirtschaft überspringen.

Sobald wir akzeptieren, dass Sprache und Vernunft nicht bloß Behelfe zur Beschreibung einer äußeren Wirklichkeit sind, sondern wichtige Instrumente der Wirklichkeitskonstruktion, dann wird auch die Bedeutung künstlicher Sprachen und technik-rationaler Formen deutlich. Wir kennen das Diktum von der *Hand als Werkzeug des Geistes*. Handlungstheoretisch und auch entwicklungsgeschichtlich gesehen wäre die Betrachtung des *Geistes als Werkzeug der Hand* ebenso gerechtfertigt. Sind es nicht die äußeren Bedingungen des Handelns, also die Strukturen der Wirklichkeit, die den Geist formen? Die Technikphilosophie hatte sehr früh auf den Parallelismus von Sprache und Werkzeug hingewiesen, dass beide das Ergebnis desselben geistigen Prinzips sind und in jedem stofflichen Werkzeug «die Kraft des Logos schlummert.»[8] Die heute dominierende Vernunftform ist gerade jene, die als Logos in die Struktur des Computers und seiner Programme eingeschrieben ist. Dort ist sie aber nicht nur implizit vorhanden, sondern als explizite Beschreibung, als operative Kraft die nicht nur Realität abbildet, sondern sie generiert. Im Computer werden formale Sprachen aktiv, sie wirken unmittelbar und realitätserzeugend.

Die Formalismen der Mathematik und der Naturwissenschaften erreichen als Codes der programmierbaren Maschine eine neue Wirkungsstufe. Formalisierung bedeutete wesentlich ***Abstraktion*** von realen Verhältnissen, also ein Zurücktreten, Generalisieren und Reinigen von für das gezielte Interesse Unwesentlichem und Zweideutigem. Mit Abstraktion wird also das bezeichnet, was an objektiv Vorhandenem abgezogen und an Ideen, also nicht Vorhandenem addiert werden muss, damit Phänomene unserer Lebenswelt zur Form werden können. Im Computer sind die eindeutigen und trennscharfen Formalismen nun aber operativ und erzeugen eigenständig neue Phänomene. In den prozessierenden Codes läuft der Abstraktionsprozess rückwärts ab, als ***Konkretisierung***. Die Interfaces, laden die sicht- und hörbar werdenden Klänge, die Bilder und Handlungen mit Mehrdeutigkeiten und Unschärfen auf. Zwar bleibt die Unterseite der Medien – der Code – streng rational, doch an der Oberfläche sind die Phänomene wieder angereichert mit Unbeabsichtigtem und

[7] Deutsche Börse, (abgerufen 13.02.2014)
https://deutsche-boerse.com/dbg/dispatch/de/kir/dbg_nav/about_us/15_Public_affairs/10_News/30_HFT
[8] Ebenda.

Nebeneffekten, die neue Interpretationsspielräume eröffnen.[9] Bei den nachfolgenden Betrachtungen steht der Unterschied zwischen zwei Formen des Konkreten im Mittelpunkt. Einmal das Konkrete, welches das Ausgangsmaterial für unsere zunächst gedanklichen, dann formalen Reproduktionen der Wirklichkeit bildet. Zum zweiten das Konkrete, das sich in der Umkehrung entfaltet, wenn formal gefasste Weltausschnitte sich in maschinellen Prozessen rekontextualiseren und rematerialisieren.

Wie eingangs erwähnt, bestehen Digitale Systeme aus Triaden, die sich für die jeweilige Prozessierung der Codes dynamisch konfigurieren:[10] 1. den *Rechnernetzen*, die noch offen (programmierbar) für die Einschreibung eines konkreten Verhaltens sind, 2. den *hybriden Interfaces*, die eingangsseitig Ereignisse der nicht-symbolischen Umwelt in Zeichen und ausgangsseitig Zeichen in Ereignisse und Handlungen umwandeln, und 3. den *Codes*, die sich als Signale in die Hardware der Rechner einschreiben und das Verhalten des Ensembles strukturieren. Um die neuen Erscheinungsweisen computerbasierter Medien zu verstehen, reicht es nicht, nur die Codes oder die Struktur der Programmiersprachen zu analysieren. Die übliche Charakterisierung von Rechenprozessen als ein formales, von der materiellen Welt abgelöstes, syntaktisches Spiel der Algorithmen greift ins Leere, wenn wir die Phänomene unserer digitalen Kultur in ihren allseitigen Grundbedingungen verstehen wollen. Auch die Materialität der Rechnernetze – mit ihren elementaren Funktionen des Speichern, Rechnens und Übertragens – und die Interfaces wirken am Gelingen der Anwendung und insbesondere auch der Ästhetik ihrer Inhalte mit.

«Mein teurer Freund, ich rat Euch drum
zuerst Collegium Logicum.
Da wird der Geist Euch wohl dressiert,
in spanische Stiefeln eingeschnürt,
dass er bedächtiger so fortan
hinschleiche die Gedankenbahn
und nicht etwa, die Kreuz und Quer,
irrlichteliere hin und her.»
Mephistopheles[11]

ABSTRAKTION - CODE ALS FORM

Was meinen wir mit Form? Hartmut Winkler weist darauf hin, dass der Begriff der Form in unterschiedlichen Medien unterschiedlich bestimmt ist. Wir folgen hier seinem Vorschlag, Form als Struktur zu denken und Formalisierung als historischen Versuch, eine Sprache für die Darstellung von Strukturen zu finden.[12] Dieser Formbegriff hat den Vorteil, dass er eine Brücke zur realen Welt schlägt indem er Programme als Strukturentwürfe auffasst, die Zusammenhänge innerhalb der Realität modellieren oder gestaltend in diese eingreifen. Mit Form meinen wir in unserem Zusammenhang also den inneren Aufbau von Codes; das, was von den Zeichensystemen überbleibt, wenn wir nicht nur ihre Materialität abziehen, sondern auch von willkürlichen Entscheidungen abstrahieren, wie der Benennung und dem konkreten Aussehen der Zeichen. Es spielt für die Form eines Algorithmus keine Rolle, welche Syntax, Variablennamen oder Schlüsselwörter wir benutzen. Als Struktur – dies ist

[9] Zur doppelten Existenz medialer Inhalte und der Unterscheidung zwischen 'Oberfläche' und 'Unterfläche' siehe auch Frieder Nake, Das Doppelte Bild, in: Horst Bredekamp, Matthias Bruhn, Gabriele Werner (Herausgeber), *Bildwelten des Wissens*, Akademie-Verlag 2006, S. 40-50.
[10] Das Wissen um den generellen Aufbau digitaler Rechensysteme ist für das Verständnis unserer nachfolgenden Überlegungen zentral, deshalb ist die Dreiteilung der Maschine im Anhang ein wenig ausführlicher beschrieben.
[11] Johann Wolfgang von Goethe, Faust 1, Studierzimmer.
[12] Hartmut Winkler, *Diskursökonomie – Versuch über die innere Ökonomie der Medien*, Suhrkamp Verlag, Frankfurt am Main 2004, S. 147ff.

der idealisierte Kern der Formalisierung – bleibt nur das Spiel der Differenzen, das sich auf die Unterscheidbarkeit der Zeichen und ihre inneren Relationen bezieht. Zwischen der formalen Struktur und dem modellierten Ausschnitt der Welt wird bei gelungener Formalisierung eine Strukturähnlichkeit behauptet. Aber erst durch die Rückbindung der Codes in einen Anwendungskontext, der über geeignete Schnittstellen während der Ausführung realisiert ist, kann der Zusammenhang zur Welt wieder hergestellt werden.

Sobald wir ein mathematisches Zeichen oder eine algebraische Struktur verwenden, hat die Ablösung von realen Verhältnissen bereits stattgefunden. In den Formalisierungen, zum Beispiel einem von Hand geschriebenen Zeichen *A*, laufen immer Idealisierungen mit, die sich bei genauerem Hinsehen als problematisch erweisen. Um ein auf Papier geschriebenes Gebilde als *A* zu identifizieren, ist nicht nur eine trainierte Wahrnehmung notwendig, die Behauptung der Gleichheit ist auch ein Willensakt: „Ja, ich akzeptiere diese der Helligkeitswerte als ein *A*". Als weitere idealisierte Voraussetzung gehen wir davon aus, dass uns unendlich viele solcher Instanzen eines *A*'s zur Verfügung stehen oder wir beliebig viele davon erzeugen können. Solche und ähnliche Idealisierungen laufen im Formbegriff für Codes immer mit. «Die Entwicklung von Formsprachen muss verstanden werden als Versuch, Form anzuschreiben, und zwar in einer möglichst expliziten Art und Weise. Computer, dies ist mein Deutungsvorschlag zur Geschichte der Formalisierung, bieten Form 'skelettiert'; unter Verzicht auf das 'Fleisch', das andere Medien brauchen, um Form überhaupt zur Erscheinung zu bringen, und das, will man die Perspektive des neuen Mediums einnehmen, als überflüssiges 'Material' erscheint. [...] Für den Begriff der 'Form' aber bedeutet dies, dass er schwebt. Massefrei leicht über dem Schmutz der dreidimensionalen Welt, in einer eigenen Sphäre, in der die Formen und Formalisierung 'autark' sind in ihrer Eigenlogik und inneren Stimmigkeit.»[13] Das wesentliche Rüstzeug um formale Strukturen hervorzubringen, ist die *Abstraktion*, die aber immer auch Idealisierung bedeutet. Formalisierung heißt deshalb nicht nur Detail- und Materialverzicht, sondern auch das Hinzufügen von Luftgespinsten. Die extreme Reduktion führt zu etwas Eigenem, das in einer materiellen Welt nicht mehr eingelöst werden kann. In der Geometrie sind das Punkte ohne Ausdehnung, unendlich lange Geraden und Flächen ohne Dicke und vieles mehr.

Wie kommt die Geometrie in die Maschine? – Ein ausführliches Beispiel

«*Die Tatsache, dass eine Wissenschaft von der Art bestehen und in der Weise aufgebaut werden kann, wie es bei der Geometrie der Fall ist, hat von jeher die Aufmerksamkeit aller Derer, welche für die prinzipiellen Fragen der Erkenntnistheorie Interesse fühlten, im höchsten Grade in Anspruch nehmen müssen. [...] Dabei fällt ihr in keiner Weise die mühsame und langwierige Aufgabe zu, Erfahrungstatsachen sammeln zu müssen, wie es die Naturwissenschaften im engeren Sinne zu tun haben, sondern die ausschließliche Form ihres wissenschaftlichen Verfahrens ist die Deduktion. Schluss wird aus Schluss entwickelt, und doch zweifelt schließlich Niemand von gesunden Sinnen daran, dass diese geometrischen Sätze ihre sehr praktische Anwendung auf die uns umgebende Wirklichkeit finden müssen. Die Feldmesskunst wie die Architektur, die Maschinenbaukunst wie die mathematische Physik, sie berechnen fortdauernd Raumverhältnisse der verschiedensten Art nach geometrischen Sätzen; sie erwarten, dass der Erfolg ihrer Konstruktionen und Versuche sich diesen Rechnungen füge, und noch ist kein Fall bekannt geworden, wo sie sich in dieser Erwartung getäuscht hätten, vorausgesetzt, dass sie richtig und mit ausreichenden Daten gerechnet hatten.*»[14]

Hermann von Helmholtz 1870

[13] Ebenda, S. 153ff.

[14] Über den Ursprung und die Bedeutung der geometrischen Axiome, Vortrag gehalten im Dozentenverein zu Heidelberg 1870 von Hermann von Helmholtz. S. 1–31 aus Helmholtz, Herrmann: Vorträge und Reden, Braunschweig, Vieweg Bd. 2, 4. Aufl. – 1883.

In der Mathematikgeschichte nehmen die *Elemente Euklids*, und dabei insbesondere die Kapitel über die Geometrie, eine herausragende Stellung ein. Die Leistungen Euklids liegen dabei weniger in Entwicklung neuer mathematischer Beweise, als vielmehr in der Systematisierung des in der Antike bereits Bekannten. Er konnte auf eine umfangreiche Tradition des mathematischen Beweisens zurückgreifen, präsentiert diese aber in der Form einer neuartigen *Axiomatischen Theorie*. Die prägende Wirkung Euklids geht von hier, vom axiomatischen Denken aus, das sich «seit dem Ende des 19. Jahrhunderts zu einer Art Weltherrschaft»[15] aufschwang. Am Anfang jeder axiomatischen Theorie stehen gewisse *Axiome* die letztlich unbewiesen bleiben. Alle Sätze und Konstruktionsverfahren der Geometrie dürfen nur unter Zuhilfenahme der Definitionen, Axiome, Postulate sowie bereits vorher gezeigter Sätze und Konstruktionen bewiesen und hergeleitet werden, und nicht etwa durch einen Hinweis auf die Überzeugungskraft der Anschauung. Auch wenn Euklids *Elemente* aus heutiger Sicht nicht die Anforderungen an eine axiomatische Theorie erfüllen, findet sich in der Stringenz der Euklidischen Methode auch ein erster Hinweis, warum sich solche deduktiven Verfahren für die Behandlung mit dem Computer eignen.[16] Die Definitionen von *Punkt*, *Linie*, *Gerade*, *Ebene* etc. stellen bei Euklid den Bezug zur vertrauten Erfahrungswelt her, während seine Postulate bereits Forderungen an die Konstruierbarkeit geometrischer Figuren enthalten. Die eigentlichen Axiome sind bei Euklid weniger geometrische als vielmehr logische Prinzipien. Sobald nun diese Grundelemente und Grundbeziehungen festgelegt sind, können Schritt für Schritt geometrische Operationen durchgeführt werden, die ihre Verbindung zur physikalischen Wirklichkeit scheinbar wie von selbst halten. Hermann von Helmholtz hakt an dieser Stelle ein. «Die Frage, welche sich mir dabei aufdrängt und die auch offenbar in das Bereich der exakten Wissenschaften gehört, war zunächst nur die: Wieviel von den Sätzen der Geometrie hat objektiv gültigen Sinn? Wieviel ist im Gegenteil nur Definition oder Folge aus Definitionen, oder von der Form der Darstellung abhängig? Diese Frage ist meines Erachtens nicht so ganz einfach zu beantworten, da wir es in der Geometrie stets mit idealen Gebilden zu tun haben, deren körperliche Darstellung in der Wirklichkeit immer nur eine Annäherung an die Forderungen des Begriffes ist, und wir darüber, ob ein Körper fest, ob seine Flächen eben, seine Kanten gerade sind, erst mittels derselben Sätze entscheiden, deren tatsächliche Richtigkeit durch die Prüfung zu erweisen wäre.»[17]

Euklids Axiome galten noch als unmittelbar einleuchtende Tatsachen und wahre Aussagen über existierende Dinge. Aber bereits für die Euklidischen Definitionen gilt: Punkte, Geraden und Ebenen existieren nirgends in der Welt, wir Denken diese Dinge und erst indem wir sie denken werden sie existent. Sobald sie aber gedacht sind, können wir sie messend und konstruierend in der Welt aufsuchen. Bei David Hilbert – gut 2000 Jahre nach Euklid – stammen die Definitionen der Axiome zwar immer noch aus konkreten Vorstellungen, in der Folge geht es ihm aber vor allem um Fragen der Vollständigkeit, Widerspruchsfreiheit und Entscheidbarkeit seiner mathematischen Strukturen, unabhängig von irgendwelchen Tatsächlichkeiten im Physischen. Diese Fragen betreffen nur noch innere Angelegenheiten der Formalismen. David Hilberts Arbeit zu den *Grundlagen der Geometrie* von 1899 beginnt mit dem Satz: «Wir denken drei verschiedene Systeme von Dingen: die Dinge des ersten Systems nennen wir *Punkte* und bezeichnen sie mit *A, B, C, …*; die Dinge des zweiten Systems nennen wir *Geraden* und bezeichnen sie mit *a, b, c, …;* die Dinge des dritten Systems nennen wir *Ebenen* und bezeichnen sie mit α, ß, γ, *…*;» In der Mathematik verselbständigen sich die formalen Systeme und entwickeln losgelöst von ihren ursprünglichen Bedeutungen eigene Wahrheiten, die nicht mehr einer äußeren Realität

[15] Wilhelm Kamlah, Paul Lorenzen, Logische Propädeutik – Vorschule des vernünftigen Redens, BI Hochschultaschenbücher, Band 227, 2. Auflage 1973, S. 17.
[16] Wobei zusätzlich die Forderung der Operationalisierbarkeit der Theorie gewährleistet sein muss, worauf zurückzukommen sein wird.
[17] H. v. Helmholtz, Über die Tatsachen, die der Geometrie zugrunde liegen, in: Nachrichten von der Königlichen Gesellschaft der Wissenschaften zu Göttingen 1868, Wiss. Abh. Bd. 2. S. 618.

gegenüber Rechenschaft ablegen, sondern nur noch ihre inneren Beziehungen betreffen.

Die Axiome der Mathematik haben heute also eine andere Bedeutung. «Freilich gelten Axiome nun nicht mehr als „evident" oder gar als Prinzipien der natürlichen Vernunft selbst, die Gott allen seinen Geschöpfen mitgegeben hat. Axiome werden vielmehr wie in freiem Entwurf „erst einmal Hingeschrieben" und nur danach beurteilt, was sie als Prämissen eines Systems von weiteren Sätzen leisten, die nach den Regeln der Logik aus ihnen hervorgehen. Auch in den exakten Wissenschaften hat man also die Frage nach einer von Anfang an überzeugenden ersten Begründung als ein Bestandsstück antiquierter Tradition abgeschrieben.»[18] Was für die Naturwissenschaften und die Philosophie gilt, trifft auch auf die Informatik zu. Es geht nicht darum, einen festen, unerschütterlichen Grund zu finden, auf dem unangreifbar das gesamte Weltgebäude als Software errichtet werden kann, sondern darum, Kalküle oder Formalisierungen zu finden, die im Hinblick auf die pragmatischen Ziele der Applikation das Notwendige leisten. Entscheidungen, die von Algorithmen getroffen werden, sind immer dessen innerem Aufbau geschuldet. Ihre Verbindung zur Welt müssen sich in konkreten Umwelt-System-Konstellationen, für die das Programm vorher vielleicht noch nie getestet wurde, erst bewähren. Wir können für Software, die in reales Geschehen eingreift, zwar sicherstellen, dass bestimmte formale Spezifikationen erfüllt sind, doch die Außenwelt, in die unsere Software eingebettet ist, lässt sich nicht vollständig spezifizieren. Es ist nicht gesichert, dass Berechnungen beim Rückübersetzen in die Welt zu 'angemessenen' Interpretationen und Handlungen oder zum Konflikt zwischen physischer und formaler Realität führen. Solches Zusammenspiel des Abstrakten mit dem Konkreten, des Geistigen mit dem Materiellen, ist auch die Grundfigur der praktischen Geometrie.

Die Frage „*Wie kommt die Geometrie in die Maschine?*" setzt also zu spät an. Es ist zuvorderst zu fragen, wie es zur Geometrie selbst kommt. Geometrie und programmierbare Maschine haben die gleichen Wurzeln: die Abstraktion, Dekontextualisierung und Schematisierung von Abläufen. Das Prinzip *abstrakter Ideen* wird in der programmierbaren Maschine nur konsequent zu Ende geführt. Um zu den Bedingungen der maschinellen Behandlung der Geometrie zu kommen, hilft es nicht, schon von den Euklidschen Definitionen des Punktes, der Linie und der Fläche auszugehen, wir müssen uns über die Entstehung dieser Begriffe klar werden.

Doch wo beginnen? Notwendigerweise bevor die Dinge zu Zeichen werden. Umberto Eco wählt in seiner *Einführung in die Semiotik*[19] die Situation eines Urzeitmenschen um über den Anfang der Architektur und aller Semiotik nachzudenken. In seiner hypothetischen Erzählung flüchtet ein Mensch der Steinzeit vor einem Unwetter in eine Höhle. Geschützt vor Kälte und Regen beginnt er ihre Konturen zu erforschen. Schemenhaft erkennt er ihre Weite und Wölbung und den Eingang als Grenze zwischen dem dunklen, schützenden Innenraum (der vielleicht undeutliche Uterus-Sehnsüchte weckt) und der hellen und unwirtlichen Außenwelt. Diese Erfahrung von *Innen* und *Außen* ermöglicht ihm allgemeine Merkmale zu identifizieren, die ihm helfen andere Höhlen zu erkennen, in denen er Schutz finden kann. Im Laufe der Zeit wird er dann die Vielzahl ähnlicher Orte durch die abstrakte *Idee der Höhle* ersetzen. «Ein Modell, eine Struktur, etwas real nicht Existierendes aufgrund dessen er aber einen bestimmten Kontext von Phänomenen als „Höhle" erkennt.»[20] Das Explizieren dieser allgemeinen Merkmale führt zum Begriff der Höhle und ist der erste Schritt in Richtung Formalisierung.

Die Herausarbeitung von Begriffen ist auch der Schlüssel zur Geometrie und zur Mathematik allgemein. Eine scheinbar einfache Aussage wie „Dies ist ein Kreis" setzt bereits enormes Können voraus. Wir müssen in der Lage sein, unsere Vorstellung von einem Kreis aus dem visuellen Stimulus zu extrahieren und unsere Sprache muss Wörter bereitstellen, die wir von unseren Eltern durch wiederholte Einübung kennen und gebrauchen gelernt

[18] Ebenda, S18.
[19] Vgl. Umberto Eco, Einführung in die Semiotik, UTB für Wissenschaft 1994, S. 296 ff.
[20] A.a.O.

haben. In unserer Kindheit erwerben wir die Unterscheidung der Dinge und der zugehörigen Wörter simultan. So trennen sich nicht nur die visuellen Erscheinungen von *Kreis* und *Dreieck*, sondern gleichzeitig das Wort 'Kreis' vom Wort 'Dreieck'. Und jede dieser Unterscheidungen schließt immer auch schon Verneinungen mit ein. Dies ist ein Kreis, es ist also kein Dreieck und es ist auch nicht Kreis und Dreieck zugleich und genauso wenig ist es in diesem Moment ein Kreis und im nächsten ein Dreieck. Das zumindest sagt uns unsere normale (vormediale) Erfahrung mit starren Körpern. Solche Tatsachen, die wir in der Interaktion mit der Welt erst selbst herstellen, sind auch der Ursprung jeder Logik. Beim Erlernen des Begriffs vollzieht sich auch bereits die Trennung von Form und Material. Die Materialität, in der uns ein Kreis gezeigt wird, ist irrelevant für unsere Entscheidung, ob es sich tatsächlich um einen Kreis handelt oder nicht. Man beachte, wir stellen diese Tatsachen her!

Die Idealisierungen, die in Euklids Definitionen des Punktes, der Linie, der Geraden und der Ebene auftauchen, lassen sich aus der Handlungspraxis heraus nachvollziehen. Die Landvermessung und das Bauwesen machten geometrische Überlegungen erforderlich. Abstände, Winkel, Geraden, Vielecke und Körper mussten nicht nur als abstrakte Vorstellungen erfunden werden, sie mussten vor allem als Handlungselemente konstruierbar sein. Genauso wie in Umberto Eco's Beispiel der Begriff der Höhle an Exempeln erlernt wird, müssen auch die Ideen der Ebene, der Kugel, des Kreises, der Vielecke, der Geraden und des Punktes im praktischen Umgang erlernt werden. Warum sie sich als Begriffe festigen konnten hat mit Handlungsaufgaben zu tun und mit den Instrumenten, die zu deren Lösung geschaffen wurden. Peter Janich weißt ausdrücklich darauf hin, dass die Elemente der Geometrie ihren Ursprung in der handwerklichen Herstellung haben.[21] Für ihn war schon Euklid als *Mundwerker* überführt, als Redefreund, «der in seiner Theorie ihren handwerklichen Ursprüngen bereits aus Unkenntnis den Rücken gekehrt hat».[22] Am Beispiel der Kugel zeigt Janich, dass Euklid die Definition der Kugel nicht aus theoretischer Analyse, sondern aus handwerklicher Fertigkeit ableitet. Euklid bestimmt den Kreis im 1. Buch, Definition 15 wie folgt: «Ein Kreis ist eine ebene, von einer einzigen Linie umfasste Figur mit der Eigenschaft, dass alle von einem innerhalb der Figur gelegenen Punkte bis zur Linie laufende Strecken einander gleich sind.» Die Definition folgt dem Werkzeug zu seiner Herstellung (dem Zirkel oder einer einfachen Leiste mit zwei Nägeln) und nicht umgekehrt. Es werden nicht Werkzeuge erfunden, um theoretische Gebilde zu realisieren, sondern umgekehrt, an den Problemen, Werkzeugen und Handlungen bilden sich die Begriffe. Ganz unhandwerklich gedacht und in direkter Erweiterung des Kreises, ist die Kugel als Fläche zu definieren, deren Punkte von einem innerhalb der Figur gelegenen Mittelpunkt gleichweit entfernt sind. Doch im 11. Buch, Definition 14, wird die Kugel definiert als «Körper, der umschlossen wird, wenn ein Halbkreis, während sein Durchmesser fest bleibt, durch Herumführen wieder in die gleiche Lage zurückgebracht wird, von der er ausging.»[23] Diese Definition ist der Schablone des Steinmetzes abgeschaut, der diese in der beschriebenen Weise benutzt, um die Kugelform zu kontrollieren. Für die algorithmische Behandlung der Kugel im Computer erweisen sich aber durchaus andere als die handwerkliche Definition als günstig.

Die Geometrie nahm ihren Anfang als Werkzeug der Praxis. Im Kern geht es um die Erweiterung menschlicher Handlungsmöglichkeiten. Die erste und zentrale Vorstellung, die vor der geometrischen Handlung existieren muss, ist die des Abstandes. Als *Längenmaß* bezeichnen wir in diesem Zusammenhang sowohl den Abstand (Strecke) zwischen zwei Orten im Raum, als auch das Objekt, welches eine bestimmte Längeneinheit verkörpert. Im Hinblick auf die Erweiterung menschlicher Handlungsmöglichkeiten ist entscheidend, dass Abstände im Raum sich auf einen starren Körper durch das Übertragen von Markierungen

[21] Peter Janich, Handwerker und Mundwerker, in: die Hand – Werkzeug des Geistes, Marco Wehr / Martin Weinmann (Hrsg.), Spektrum Akademischer Verlag 1999, S. 274.
[22] Ebenda S. 278.
[23] Peter Janich, a.a.O.

kopieren lassen. Jede Strecke, so die Grundidee, lässt sich damit an beliebige Orte im Raum transportieren. Der Begriff des Abstands, als gedachte Verbindung zwischen den zwei Markierungen, führt auch direkt zur Idee der Geraden. Nicht die Form des Objektes das die Markierungen trägt ist wichtig, es zählt nur die kürzeste Verbindung zwischen den Markierungspunkten. Eine gespannte Schnur zwischen zwei Pflöcken kommt der Vorstellung sehr nahe. Wir vollziehen Handlungen an einer abstrakten Eigenschaft des Raums, es geht nicht darum das Längenmaß als Objekt zu bewegen, sondern die damit verbundene Idee wird bewegt.

Der erste Nachweis der Verwendung einer Maßeinheit findet sich schon bei Bandkeramischen Häusern 5000 vor Christus und der erste dingliche Maßstab wurde bei Ausgrabungen im Tempel E von Nippur in Mesopotamien entdeckt und inzwischen in die erste Hälfte des 3. Jahrtausends v. Chr. datiert.[24] Viele vormetrische Längeneinheiten orientierten sich an menschlichen Körpermaßen (Klafter, Elle, Fuß, Spanne, Digitus, etc.), was daran liegen mag, dass man diese Maße immer sofort zur Verfügung standen und es bei den vielen Anwendungen nicht auf die absolute Maßeinheit ankommt, sondern ad hoc eine Länge in Relation zu einer anderen gesetzt werden muss. Schon bei einfachen baulichen Tätigkeiten erweist es sich oft als notwendig, dass Abstände und Längen nicht nur grob abgeschätzt, sondern relativ präzise von einem Gegenstand auf einen anderen übertragen werden können. Sobald man aber daran interessiert ist, die Übertragung von Strecken immer präziser zu gestalten, ergibt sich zwangsweise, dass die Markierungspunkte immer enger eingegrenzt werden müssen, bis sie schließlich in der Vorstellung zu unendlich kleinen Punkten schrumpfen. Auch die unvermeidliche Breite realer Linien ist für präzise Konstruktionen nur störend. Die Logik des Konstruierens braucht keine Linienstärke, lediglich für die Sichtbarkeit des Konstruierten ist sie notwendig. Die Welt, in der diese geometrischen Konstruktionen stattfinden, ist die Ebene. Mit dem Begriff '*eben*' wird eine weiteres Ideal eingeführt. Auch diese Leitfigur leitet sich aus handwerklichen Erfahrungen ab. Ein Streit, ob eine Fläche wirklich eben ist, lässt sich nicht sprachlich, sondern nur durch Handlung entscheiden. Ein auf der Fläche herumgeführtes Lineal darf an keiner Stelle eine Lücke zwischen Fläche und Lineal aufweisen, wo auch immer das Lineal angelegt wird. Eine andere Methode die Eigenschaft 'eben' zu testen besteht darin, drei Flächen paarweise aufeinander zu legen, wobei sich auch hier nirgends ein Zwischenraum zeigen darf. Sobald man solche *Störungen* gedanklich immer weiter verringert, gelangt man zum Begriff der Ebene. Die geometrischen Idealisierungen sind also direkte Folge des Strebens nach Exaktheit. Dabei ist nicht entscheidend, dass wir uns den idealen Punkt, die ideale Linie oder Fläche herstellen können, ausschlaggebend ist, dass der gedankliche Prozess der dahinter steht, an keiner Stelle abgebrochen werden kann. Solange der Punkt noch eine Ausdehnung, die Linie noch eine Stärke und die Ebene noch eine Störung hat, können wir gedanklich auch noch etwas abziehen und verbessern. Der Kern der Abstraktion ist dieses gedankliche immer weiter machen können, daraus folgt zwangsläufig das nicht mehr zur Welt gehörende Ideal, das wir unseren Handlungen aber zugrunde legen.

Auf der Idee der Übertragung von Strecken im Raum baut die gesamte Euklidische Geometrie auf. In ihrer klassischen Ausprägung kennt sie kein Maß, sondern nur Verhältnisse. Als Werkzeuge der Konstruktion sind bei Euklid deshalb nur Zirkel und Lineal zugelassen, wobei das Lineal keine Markierungen hat und sich somit auch keine Messungen vornehmen lassen. Aus der Idee der Ebene und den beiden idealisierten Euklidischen Werkzeugen, mit denen wir in der Vorstellung die Geraden unendlich präzise ziehen und Punkte unendlich genau markieren können, leitet sich das gesamte Gebäude der geometrischen Gesetzmäßigkeiten ab. Sobald der künstliche Handlungsrahmen fixiert und die Ungenauigkeit der physischen Welt beseitigt sind, können wir das Universum der möglichen Konstruktionen daraus entfalten.

[24] Rolf C. Rottländer, Rottenburg / Köln, http://vormetrische-laengeneinheiten.de (abgerufen 7. Feb. 2014).

Eine Geometrie, die ohne Zahlen auskommt und wie oben beschrieben auf Axiomen und Konstruktionsmethoden basiert, wird auch *Synthetische Geometrie* genannt. Rechner dagegen arbeiten mit Zahlensystemen. Geometrie ist deshalb nur dann mit dem Computer behandelbar, wenn geometrische Verhältnisse und Aufgabenstellungen in die Welt der Zahlen übersetzt werden. Das entscheidende Verbindungsglied zwischen Geometrie und Rechenoperationen bilden Koordinatensysteme. Sie erlauben es, geometrische Beziehungen durch algebraische Gleichungen zu beschreiben. Als Erfinder dieser *Analytischen Geometrie* gilt der französische Philosoph, Mathematiker und Naturwissenschaftler Rene Descartes (1596 – 1650). Ihm zu Ehren nennt man Koordinatensysteme mit paarweise senkrecht aufeinander stehenden Achsen und gleichmäßiger Skaleneinteilung auch *kartesische Koordinatensysteme*. In der modernen Mathematik werden die Rechnungen der analytischen Geometrie durch die Vektorrechnung vereinheitlicht. Die Darstellung von Punkten im Raum durch Vektoren im kartesischen Koordinatensystem erscheint sogar Anfängern als natürliche und leicht nachzuvollziehende Darstellung. Zusammen mit den mathematischen Konzepten der *Funktion*[25] und der *Variablen* gelingt die vollständige Übersetzung der *Synthetischen Geometrie* in die *Analytische* [siehe Anhang B].

Vektorrechnung kann als allgemeine Zuordnungsvorschrift – als Funktion von mathematischen Objekten – aufgefasst werden. Durch Funktionen werden nicht mehr einzelne Beziehungen zwischen Elementen notiert, sondern zwischen ganzen Mengen. Jedem Element einer Ausgangsmenge wird funktional, d.h. rechnerisch ein Element der Zielmenge zugeordnet. Wir müssen nicht für alle Elemente der Ausgangsmenge auflisten, mit welchem Zielelement sie verbunden sind, sondern nur die Zuordnungsvorschrift, d.h. den entsprechenden *Operator* implementieren. Der entscheidende Kniff, der dann die Mächtigkeit funktionaler Beschreibungen herstellt, sind die *Variablen*[26]. Mit der Einführung der Algebra, d.h. von Symbolen für *Rechenoperationen* und *Variablen*, ist auch die direkte Verbindung zur programmierbaren Maschine hergestellt. Mit der Algebra stehen Bezeichner zur Verfügung, die für ein beliebiges Element aus einer vorgegebenen Menge stehen können. Der Wert, den eine Variable im Rahmen einer konkreten Rechnung annimmt, kann auf diese Weise über den Namen der Variable adressiert werden.

In der Programmierung wird mit Variablen die Vorstellung von Behältern verbunden, die eine Rechengröße oder ein Zeichen aus einer bestimmten Vorratsmenge aufnehmen können. Für die Variable wird im Programm ein Name vergeben der mit einer bestimmten Adresse im Speicher der Maschine verbunden ist. Die Werte in den Speicherzellen können durch elektronische oder auf irgendeine andere Weise realisierte Operationen verändert werden. Probleme, die formalisiert und in operationaler Beschreibung vorliegen, werden durch die technische Realisierung von Operatoren und Speicherzellen maschinell lösbar. Dabei ist deren konkrete materielle Umsetzung den eigentlichen Algorithmen und Formalismen gegenüber nur äußerlich. Das Material in dem die Operationen und Variablen gebaut sind, ist unbedeutend, entscheidend ist die Funktion die realisiert wird. Durch den mathematischen Kunstgriff der *Variablen* und ihrer maschinellen Realisierung als adressierbare und mittels Operatoren veränderliche Speicherzellen wird eine enorme Flexibilisierung von automatisierten Handlungsmöglichkeiten erreicht. Maschinell realisierte Variablen sind durch vier Aspekte gekennzeichnet: 1. dem Speicherplatz als Behältnis, 2. dem im Speicherplatz abgelegten Datum, 3. der Adresse des Speicherplatzes in der Maschine und 4. dem Namen, unter dem der Speicherplatz angesprochen werden kann. Damit sind die wichtigsten Verbindungen zwischen Formalismus und Maschine charakterisiert. Der Programmierer stellt durch das Programm eine Beziehung zwischen

[25] Auch als Abbildung oder Zuordnungsvorschrift bezeichnet.

[26] Die Verwendung von Buchstaben als Variablen in mathematischen Notationen stammt von François Viète (lat. Franciscus Vieta, 1540 -1603). Vieta unterscheidet das Zahlenrechnen, die logistica numerosa, vom Buchstabenrechnen, der logistica speciosa und gilt damit als Begründer der modernen Algebra. Statt von Variablen spricht die Mathematik auch von *Platzhaltern* oder *Veränderlichen*.

der materiellen Maschine und den formalen Beschreibungen her. Der erfahrene Entwickler ist in der Lage, in seiner Betrachtungsweise zwischen der semiotischen Ebene seiner Zeichenmanipulationen, der Vorstellung von ihrer technischen Realisierung durch die Maschine und den materiellen Eigenschaften seiner Interfaces umzuschalten. Was er nicht kann, ist das Ergebnis schon in seiner ganzen Fülle zu antizipieren.

Damit ist der gesamte Weg, ausgehend von einem fiktiven geometrischen Problem – etwa der Konstruktion einer geometrischen Figur auf einem Blatt Papier mit Hilfe von Zirkel und Lineal – hin zur Darstellung und algorithmischen Behandlung dieses Problems in der Maschine in groben Zügen skizziert.[27] In Anhang B wird der beschriebene Weg vom geometrischen Problem zum Computerprogramm an einem Beispiel nachvollzogen. Bevor wir uns für den Rest des Artikels vollständig der Materialseite des Rechnens zuwenden, wollen wir abschließend einige Kennzeichen der Formalisierung und Algorithmisierung zusammenfassen:

- *Verallgemeinerung*: Algorithmen behandeln nicht ein einzelnes Problem, sondern immer eine Klasse von Problemen. Wir implementieren keinen Algorithmus, der uns das Ergebnis von 17x23 berechnet, sondern das Problem der Multiplikation löst. Das algorithmische Denken ist immer ein Denken in solchen Problemklassen. Nach der Implementierung eines Programms werden die konkreten Instanzen einer Problemklasse autonom von der Maschine bearbeitet. Durch die formale Methodik, die eine automatisierte Behandlung der Aufgabe mit der Maschine ermöglicht, wird unreflektierte Wiederholbarkeit hergestellt. Bei komplexen Aufgaben und entsprechend umfangreichen Programmstrukturen liegt darin allerdings auch eine der Hauptgefahren automatisierter Vorgänge. In komplexen Anwendungsbereichen gibt es immer Konstellationen, wo die dekontextualisierten Algorithmen versagen oder zumindest unerwünschte Nebenwirkungen im Realen produzieren.
- *Verselbständigung*: Die formalen Algorithmen sind inhaltslos. Die inhaltliche Bindung entsteht erst durch die Einbettung in eine reale Umwelt. Algorithmen der Signalverarbeitung können beispielsweise in gleicher Weise auf Bilder angewandt werden wie auf Töne oder Bewegungen. Wenn die inhaltliche Bindung an den Kontext etwas zu lösendes bzw. herstellbares ist, muss den Algorithmen eine eigene Realität zugesprochen werden. Eine Realität, die unabhängig von konkreten inhaltlichen Bindungen untersucht werden kann. Was wir dabei untersuchen sind ihre inneren Strukturgesetze. Die stärksten Computeranwendungen entstehen oft dort, wo eine fundierte, gut ausgearbeitete Theorie schon für die Implementierung bereitsteht. Programme sind in diesem Fall tatsächlich *implementierte Theorie*[28].
- *Synthese*: Durch die Einbettung der Programme in ein Wirkungsumfeld entsteht immer schon neues, auch wenn der Modellierung die Analyse realer Verhältnisse zugrunde liegt. Die Abbildfunktion, wie wir sie für die Geometrie beschrieben haben, kann aber auch von vorne herein fallengelassen werden. Wir bewegen uns dann nicht mehr auf der Seite der Weltanalyse und der Rekonstruktion, sondern der Weltsynthese. Algorithmen müssen nicht Verhältnisse unserer Erfahrungen abbilden, sie können aufgrund ihrer inneren Struktur und der Einbindung in einen lebensweltlichen Zusammenhang vorbildlos neue Sachverhalte erzeugen. Das heißt, wir können Prozesse 'materialisieren', die vorher keine Entsprechung in unserer Erfahrungswelt hatten.

[27] Es ging hier vor allem darum, einige grundlegende Abstraktionsschritte zu benennen und weniger um historische Vollständigkeit oder Präzision. Hierfür stehen in der Literatur zahlreiche Quellen zur Verfügung. Zum Beispiel: Sybille Krämer, Symbolische Maschinen – Die Idee der Formalisierung im geschichtlichen Abriss, Wissenschaftliche Buchgesellschaft, Darmstadt 1988. Bettina Heintz, Die Herrschaft der Regel – Zur Grundlagengeschichte des Computers, Campus Verlag 1993.

[28] Ein Begriff von Eggert Holling, Peter Kempin, a.a.O.

Die Entstehungsgründe formaler Modelle sind heute in der Mathematik vollständig zugunsten ihrer strukturellen Eigenschaften in den Hintergrund getreten. «Experimentelle Anordnung und mathematische Beschreibung erscheinen nun als einander völlig äußerlich. Bezüge zwischen einem mathematischen Modell und einem experimentellen System werden nur unter dem Gesichtspunkt der Zweckmäßigkeit hergestellt. Dieses oder jenes Phänomen des Experiments lässt sich annähernd mit dieser oder jener Formel beschreiben. Dies wird nicht mit einer inhaltlichen Beziehung begründet, sondern mit der Ähnlichkeit einer in beiden Bereichen gefundenen formalen Struktur. [...] Tatsächlich sind die meisten Abstraktionen, wie sie z.B. in der Mathematik verwendet werden, aus konkreten Zusammenhängen entstanden, erst durch die relativ neue, strukturelle Betrachtungsweise wird dieser Entstehungszusammenhang ausgelöscht. Sie haben nun eine autonome Existenz, die konkreten Bedingungen ihres Entstehens sind uninteressant geworden.»[29] Die Abstraktion ist damit auf die Spitze getrieben, die formalen Strukturen haben jede inhaltliche Referenz und jeden Kontextbezug abgestreift. Dem Verlust des Kontextes stehen aber beträchtliche Gewinne gegenüber. Die Untersuchung formaler Strukturen als eigener Gegenstand kann damit überhaupt erst beginnen.

> *«Die Mathematik ist eine gar herrliche Wissenschaft, aber die Mathematiker taugen oft den Henker nicht. ... so verlangt sehr oft der so genannte Mathematiker für einen tiefen Denker gehalten zu werden, ob es gleich darunter die größten Plunderköpfe gibt, die man nur finden kann, untauglich zu irgendeinem Geschäft, das Nachdenken erfordert, wenn es nicht unmittelbar durch jene leichte Verbindung von Zeichen geschehen kann, die mehr das Werk der Routine sind als die des Denkens»*
>
> Georg Christoph Lichtenberg[30]

Was wir unter der Überschrift '*Abstraktion – Code als Form*' abgehandelt haben kann als formale Rationalität bezeichnet werden. Vernünftiges handeln wird hier auf die Fähigkeit der Abstraktion und des logischen Schließens, unabhängig vom Inhalt reduziert. Denken reduziert sich selbst auf einen Mechanismus. Diese Rationalität markiert in Wirklichkeit den Weg in die Unvernunft. Es wird ein gefährlicher Dualismus von Form und Inhalt installiert, der – sobald er im großen Stil in die Realwelt, auf Technologie, Wirtschaft und Politik übertragen wird – die Selbstunterwerfung unter die Herrschaft des Faktischen bedeutet. Ungleiches wird vergleichbar und alle Qualitäten eliminiert. Wenn wir unser Denken freiwillig auf abstrakte mathematische und logische Formalismen verkürzen, geht die eigentliche, aufs Ganze zielende Vernunft unweigerlich verloren. Formale Rationalität kann nur Verfügungswissen abbilden, Orientierungswissen braucht lebendiges Weltverständnis. Wichtig für unsere Formbemühungen ist also, wie und mit welchen Inhalten wir die abstrakten Extrakte verbinden. Wir müssen uns bewusst sein, was sie zu leisten vermögen und wo ihre Grenzen verlaufen. Diese Entscheidungen sollten nicht die Kalküle selbst treffen, sie liegen in unserer Verantwortung.

KONKRETISIERUNG – CODE ALS MATERIAL

Das Konkrete ist das Antonym zum Abstrakten und Abstraktion der Vorgang bei dem sich unsere Beschreibungen von den konkreten Erscheinungen lösen. Durch den Verlust der Gegenständlichkeit kann die Abstraktion dem, der den Schlüssel nicht besitzt, bedeutungslos erscheinen. Es ist also immer zu fragen, wo die Verbindung zwischen der abstrakten Form und dem konkreten Phänomen gehalten wird. Das kann in unseren Köpfen geschehen, wenn wir eine Vorstellung memorieren, wie unsere Variablen und Funktionen mit Welt verbunden sind. Auf diese Weise können wir die Wirkungen, die Programm in seiner Laufumgebung haben wird, bis zu einem gewissen Grad antizipieren. Der

[29] Ebenda., S. 88.
[30] Lichtenberg, Sudelbücher, S. 471.

Zusammenhang zwischen Code und Umwelt geht durch die Implementierung des Formalismus in das Material der programmierbaren Apparatur über. Im Zusammenhang mit programmierbaren Maschinen ist die Feststellung entscheidend, dass im Rechenprozess die vorangegangene Abstraktion umschlägt in Konkretisierung. Indem wir die operationalen, semiotischen Systeme auf materiellen Architekturen ablaufen lassen, wird das Konkrete zurückgeholt. Gleichzeitig werden Kontingenzen und unbeabsichtigte Phänomene erzeugt, die sich aus dem dynamischen Zusammenspiel des Programmcodes, der Materialität des Rechnens und der Einbettung der Rechnung in einen äußeren Kontext ergeben. Das Fleisch digitaler Medien entsteht durch die Materialität des Rechnens und die Reintegration der algorithmischen Prozesse in physische, psychische, soziale Wirklichkeiten. Diese Materialisierung und Rekontextualisierung des Formalen bezeichnen wir hier als *Konkretisierung*.

Entsprechend der im Anhang A beschriebenen Trinität digitaler Systeme zeigt sich auch im Rechenprozess eine dreifache Materialität: Die Materialität der rechnenden Netze, die der analogen messenden, darstellenden und steuernden Interfaces und die der diskreten Codes. Die Materialität der Codes fällt in der Ausführung mit den anderen beiden Materialitäten zusammen, kann aber zwischenzeitlich herausgelöst und auf andere Materialien ausgelagert werden, z.B. Papier, um die Lesbarkeit oder Dokumentierbarkeit zu verbessern. Der Code ist insofern als eigenständige materielle Einheit zu betrachten, als er nicht an eine konkrete Maschine gebunden ist. Er kann im Rechner gespeichert sein, auf einem externen Datenträger ausgelagert, oder wie im Fall unseres Beispiels in Anhang B Teil eines Artikels sein, der ausgedruckt oder am Bildschirm gelesen werden mag. Um existent zu sein, braucht es aber irgendein Material, auf dem er notiert ist, selbst wenn er nur im Gedächtnis memoriert wird. Damit daraus ein Rechenprozess wird und sich die beabsichtigte Wirkung entfaltet, wird ein Handlungsträger (Prozessor) benötigt, der ihn interpretieren und die technischen Operationen vollziehen kann. Der Code kann einerseits, je nach Programmiersprache, auf unterschiedlichste Weise notiert werden und andererseits unterschiedliche Algorithmen repräsentieren, zum Beispiel unterschiedliche Verfahren nach denen eine Gerade zu erzeugen ist. Entscheidend für unsere Betrachtungen ist, dass das dargestellte Objekt ein Zusammenspiel der Elemente erfordert, das allerdings aus einer in der Regel zeitlich befristeten, losen Konfiguration besteht. Die einzelnen Komponenten können sich im nächsten Moment schon mit anderen Komponenten verbinden um ein neues Objekt zu realisieren. Sie verbrauchen sich nicht exklusiv an eine konkrete Konfiguration aus Hardware, Interface und Code, sondern sind Elemente eines dynamischen Spiels.

Weiterhin wichtig ist, dass wir für die Konkretisierung der Geometrie auch auf der Ebene des Codes Entscheidungen treffen, die nicht nur formal-rationaler Natur sind, sondern ästhetischer. Wie stark soll die Linie der Geraden sein, welche Farben werden für Figur und Hintergrund gewählt, wie lang soll die Gerade im Verhältnis zur Gesamtfläche sein, welche Auflösung ist zu wählen? Diese ästhetischen Entscheidungen im Code werden auf die Materialität des Interfaces Rücksicht nehmen müssen. Selbst die Entscheidungen in Bezug auf den gewählten Algorithmus müssen auf die Materialität des Rechners eingehen, und zwar insofern, als Speicherplatz und zeitliches Verhalten möglicherweise Einfluss auf die Wahl des Verfahrens haben werden. Codes sind somit ein Verbundwerkstoff, der immaterielle Algorithmen mit den materiellen Interfaces zum ‘informierten Material’ vereint. In diesen Objekten treffen sich zwei Kulturen, die rationalistische Tradition, aus der sich die algorithmischen Codes entwickelt haben, und die ästhetische Tradition, die an die Interfaces und den sich dort zeigenden Inhalten gebunden ist. Programmgesteuerte Objekte herzustellen heißt, permanent beide Seiten dieser Janusgestalt digitaler Medien zu berücksichtigen und Übersetzungsarbeit in beide Richtungen zu leisten.

Im Folgenden wollen wir einige Quellen für die sich im Rechenprozess entfaltende Fülle näher betrachten.

Die Fülle des Codes

Heute können wir auch komplexe räumliche Szenerien relativ leicht mit Hilfe von 3D Software modellieren und – Beispielsweise in Computerspielen – deren interaktives Verhalten in Echtzeit berechnen und darstellen. Die anfallenden Daten und der Code der 3D Modelle, Animationen und Rendering-Algorithmen lassen sich relativ kompakt darstellen. Aus dem Quellcode kann aber niemand die visuelle Qualität, den Detailreichtum und das interaktive Verhalten des dynamischen Ablaufs ablesen. Schon eine kurze Folge von Handlungsanweisungen kann in unserer Vorstellung, ohne Unterstützung äußerer Notationshilfen, nicht mehr zu einer Bildvorstellung zusammengefügt werden. Die Absichten des Programmierers oder des *Bildkonstrukteurs* können deshalb nur in einem iterativen Prozess, bei dem der Blick zwischen Code und Bild hin und her wechselt, angenähert werden. Aus dem sichtbaren Bild muss der Konstrukteur ableiten, welche Parameter auf der Unterfläche zu verändern sind, bis sich das gewünschte Bild einstellt. Obwohl die sinnliche Qualität des Bildes erst im Interface entsteht, ist sein Detailreichtum in den Zahlenfolgen – zum Beispiel einer zugehörigen jpeg-Datei – bereits vollständig festgelegt. Hierauf soll es an dieser Stelle ankommen, auf die Bildfülle, die aus der Berechnung resultiert. Der sinnliche Reichtum der zusätzlich durch das physische Interface hinzugefügt wird, wird weiter unten behandelt. Hinter den Rendering-Algorithmen, die das Bild aus dem geometrischen Modell berechnen, stehen heute meist komplexe physikalische Methoden, die in der Lage sind, die Lichtverteilungen im Raum realitätsnah zu simulieren. Durch formale Approximationen physikalischer Gesetze und deren Implementierung kann eine ähnliche Bildfülle erreicht werden, wie mit einem Fotoapparat.

Hierbei kommt das Wesen aller Gesetzmäßigkeit zum tragen. Durch ein physikalisches Gesetz wird eine Vielzahl von Einzelphänomenen auf ein Grundprinzip reduziert und kompakt durch eine Formel ausgedrückt. Zum Beispiel ist in den Newtonschen Gesetzen[31] die Bewegung starrer Körper auf eine sehr allgemeine und komprimierte Weise abgefasst. Durch die Implementierung und algorithmische Lösung der Newtonschen Gesetze lassen sich alle Bewegungsphänomene, die wir aus unserer Erfahrung mit starren Körpern kennen, wieder '*dekomprimieren*'. Bei geeigneter Wahl des Modells und seiner Parameter können so unzählige Einzelphänomene aus der selben formalen Struktur '*entfaltet*' werden. Mit Begriffen wie *Entfaltung* und *Dekompression* versuchen wir die Umkehrung der Theoriebildung begrifflich zu fassen.[32] Jahrhunderte hat die Wissenschaft vor allem in die Untersuchung und Beschreibung der Gesetzmäßigkeiten investiert, die Anwendung der Gesetze musste weitestgehend von Hand geschehen und wurde nur durch einfache Instrumente unterstützt. Im Computer kann nun, sofern ein operationalisiertes Modell vorliegt, die Reproduktion konkreter Phänomene automatisiert ablaufen. Damit wechseln Gesetzmäßigkeiten und formale Beschreibungen gewissermaßen die Seite, vom Werkzeug der Analyse werden sie zum Werkzeug der Synthese.

Bereits sehr einfache, aus wenigen Befehlen bestehende Programme, sind in der Lage, komplexe Strukturen zu erzeugen. Es existiert eine grundsätzliche Differenz zwischen statischem Programmcode und seiner dynamischen Ausführung. Sofern wir den momentanen Zustand der Berechnung kennen, kann zwar problemlos entschieden werden, welcher Schritt als nächster auszuführen ist. Doch das Gesamtverhalten der Programme ist

[31] Exakter wäre von physikalischer Dynamik zu sprechen, hier insbesondere die Kinetik, die den Zusammenhang von Kräften und Bewegung erfasst.

[32] Da es sich um die Herausbildung von Strukturen infolge des Zusammenspiels lokaler Operationen handelt, könnte man in diesem Zusammenhang auch den Begriff *Emergenz* verwenden. Wir verzichten hier bewusst darauf, da der Begriff dem spontanen Entstehen neuer Eigenschaften vorbehalten bleiben soll. Bei Computerprogrammen entstehen nicht generell komplexe Strukturen. Aber selbst wenn sie nicht emergent sind, wird durch die Berechnung immer etwas entfaltet. Sofern wir mit *Emergenz* lediglich etwas bezeichnen, das bereits existent, aber noch verborgen ist, dann wäre der Begriff durchaus zutreffend, da Programme deterministisch wiederholbar sind und immer wieder das gleiche Entbergen. Konrad Lorenz folgend könnte man 'echte' Emergenz dann als *Fulguration* bezeichnen.

deshalb noch lange nicht in jedem Fall vorhersehbar. Wir können sehr einfache Algorithmen anschreiben, deren einzelne Rechenschritte leicht verständlich sind, von denen wir trotzdem nicht sagen können, welche semantischen Eigenschaften sie besitzen, nicht einmal ob sie halten oder immer weiter laufen. Nun könnte man naiv an die Sache gehen und sagen, wir sind nur zu langsam oder zu unpräzise um Programme nachzuvollziehen. Dieses Defizit ist aber auch nicht zu beheben, indem wir wiederum Rechenleistung zur Bearbeitung des Problems heranziehen. Eines der zentralen Ergebnisse der Berechenbarkeitstheorie zeigt, dass es grundsätzlich kein Programm geben kann, das von (allen) anderen Programmen entscheidet, ob sie eine bestimmte Eigenschaft haben oder nicht.[33] Die Goldbachsche Vermutung [siehe Anhang C] ist ein eindrückliches Beispiel das zeigt, wie wenig wir über den Verlauf von Berechnungen aussagen können, ohne die Programme laufen zu lassen. Schon eindimensionale *Zelluläre Automaten*, deren Verhalten leicht von Hand, mit Bleistift auf dem Papier aufgezeichnet werden kann, sind zur Selbstorganisation fähig. Dies ist auch die Kernaussage des Buches „*A new Kind of Science*" von Stephen Wolfram: Programme, die aus einfachen Regeln bestehen, können Verhalten von großer Komplexität erzeugen. Sehr bekannt im Zusammenhang mit der Untersuchung eindimensionaler zellulärer Automaten ist Regel 110. Der auf dieser Regel basierende Automat kann universelle Berechnungen ausführen und besitzt damit die Mächtigkeit einer Turingmaschine. Es reicht also schon eine einzige Regel, aus der sich gesamte Universum des Rechnens entfalten lässt. Was sagt das über die physikalische Wirklichkeit, über die Mathematik, über unser Denken?

Die Physis der Hardware und der Sinnesreichtum der Interfaces

> *Es ist möglich, den Effekt einer Rechenmaschine zu erreichen, indem man eine Liste von Handlungsanweisungen niederschreibt und einen Menschen bittet, sie auszuführen. Eine derartige Kombination eines Menschen mit geschriebenen Instruktionen wird ›Papiermaschine‹ genannt. Ein Mensch, ausgestattet mit Papier, Bleistift und Radiergummi sowie strikter Disziplin unterworfen, ist in der Tat eine Universalmaschine.*[34] Alan M. Turing

Turings Universalmaschine aus Fleisch und Blut ist ein geeignetes Bild, um die Materialität des Rechnens zu veranschaulichen. Der Mensch kann sich wie eine Turingmaschine verhalten. Sobald er sich der strikten Disziplin der Regeln unterwirft, ist er tatsächlich eine Maschine, gleichzeitig kann er nicht vollständig darauf reduziert werden. Schon seine bloße physische Existenz erlaubt jedem Außenstehenden Perspektiven einzunehmen, die vom Rechenvorgang vollkommen unabhängig sind. Ein Geschehen als Rechenvorgang zu interpretieren, bedeutet immer Konzentration auf einen einzigen Aspekt und Ignoranz des Restes. Der Vorgang selbst ist immer viel reichhaltiger und kann auch im Hinblick auf andere Aspekte betrachtet werden. Natürlich gibt es eine Leitperspektive, die mit der Zweckbestimmung des Apparates als Rechner zu tun hat. Diese Sichtweise erklärt sich aber nicht von selbst, sie muss erkannt werden.[35]

Unter dem Leitbegriff *Biological Computing* stoßen wir in Grenzbereiche des Rechnens vor, die eine monokontexturale Sichtweise auf die Maschine zumindest in Frage stellen.[36]

---

[33] Satz von (Henry Gordon) Rice (1951): Es sei R die Klasse aller berechenbaren Funktionen und S eine beliebige nichttriviale (das bedeutet $S \neq \emptyset$ und $S \neq R$) Teilmenge davon. Es sei außerdem eine Kodierung, die einem Codewort w das dadurch codierte Programm P_w zuordnet, vorausgesetzt. Dann ist die Sprache $C(S) = \{ w \mid$ die von P_w berechnete Funktion liegt in $S \}$ nicht entscheidbar.

[34] Alan M. Turing, Intelligent Machinery, zit. nach: Alan M. Turing, *Intelligence Service*, Hg. B. Dotzler, F. Kittler, Berlin 1987, S. 91.

[35] Die Kulturgeschichtsschreibung lehrt uns, dass die Verwendungsweise und Bedeutung gefundener Werkzeuge sich erst erschließt, wenn auch der kulturelle Kontext bekannt ist und man weiß, wie die Menschen zur Zeit der Werkzeugnutzung gelebt und gearbeitet haben.

[36] Siehe dazu auch: Georg Trogemann, Synthese von Maschine und Biologie – Organische Maschinen und die Mechanisierung des Lebens, in: Synthesis: Zur Konjunktur eines philosophischen Begriffs in

Dort werden Bakterienkolonien gezüchtet, um das *Hamilton-Pfad-Problem* zu lösen. Das Wachstum von Schleimpilzen so gesteuert, dass sie in der Lage sind, das bekannte mathematische Problem des kürzesten Pfades in einem Labyrinth zu lösen. Roboter können durch kontinuierliche Selbstmodellierung Funktionsstörungen kompensieren und auf diese Weise qualitativ neues Verhalten entwickeln. Die Biologie vergleicht Zellen in ihrem Verhalten durchaus mit Maschinen. Das funktionale Verhalten komplexerer Organismen kann in seiner Gesamtheit nicht mit einer einzigen Beschreibungsebene erklärt werden. Dass sich ein Organismus so disziplinieren oder modifizieren lässt, dass er rechnet, heißt nicht, dass man ihn vollständig auf diese Sichtweise reduzieren kann. Die Komplexität eines lebendiger Organismen wird dabei überhaupt nicht erfasst. Wir kennen die notwendigen Bedingungen und die Grenze nicht, ab der wir sagen müssen: Das *Embodiment* der Algorithmen hat eine Stufe erreicht, auf der wir unseren Maschinen ein Eigenleben nicht mehr absprechen können, auch wenn dieses nur als unerwünschte Nebenwirkung auftritt.[37] Die synthetische Biologie hat aber unzweifelhaft das Potential, die Trennung zwischen Organismus und Maschine grundsätzlich in Frage zu stellen.

Eine Maschine als Organismus zu realisieren ist der eine Weg. Der andere, der im Zuge des Zusammenwachsens von Organismus und Maschine ebenfalls erforscht wird und in der Medizin seit längerem getestet und angewendet wird, ist die Erweiterung des Organischen um technische Sinne und informationsverarbeitende Prothesen. Das bloße Ein-Ausgabeverhalten elektronischer Sensoren und Aktoren lässt sich zwar leicht formal beschreiben, doch wenn die Signale reale Empfindungen triggern, ist dieses Verhalten wenig Aussagekräftig. Wenn die ausgelösten Wahrnehmungen wesentlicher Bestandteil der Zweckbestimmung des Geräts sind, müssen sie auch Teil der Systembeschreibung sein. Um die Wirkungsweise der Apparatur zu verstehen reicht es nicht, die inneren Funktionen und die Quantitäten der Signale an den Ein- und Ausgängen zu notieren, es müssen die Wirkungen, sinnlichen Qualitäten und Empfindungen beim Träger einbezogen werden. Verkürzt lässt sich sagen: Ein Cochleaimplantat ist dann gut, wenn der Patient damit gut hört und eine Sehprothese, wenn der Patient ein zufriedenstellendes Farbspektrum, Detailreichtum und andere Qualitäten bestätigt. Die Signale an der Schnittstelle sind dem subjektiven Erleben untergeordnet. Langfristiges Ziel der Wissenschaft ist es natürlich, die Signale direkt dort abzugreifen und zur Verfügung zu stellen, wo sie verarbeitet werden, im Gehirn. Die stetig fortschreitende Miniaturisierung elektronischer Schaltkreise zusammen mit der Verfügbarkeit kleinster energieeffizienter Funksysteme hat zur Entwicklung erster integrierter Steuerungen geführt, die derzeit an Insekten getestet werden. Mit den winzigen implantierten Chips lässt sich die Flugbahn von Insekten über einen längeren Zeitraum von den Steuertasten eines Labtops aus kontrollieren.

Es gibt aber sehr viel einfachere Beispiele, mit denen wir die Materialität von Rechenprozessen und ihren Einfluss auf das Ergebnis auch bei herkömmlichen elektronischen Computern verdeutlichen können. Das fraktale Muster im unteren Dreieck von Abbildung 1 kann als materielle Störung Euklidischer Idealisierungen interpretiert werden. Zwei Dreiecke (links schwarz, rechts weiß) werden per 3D-Software[38] so übereinander gelegt, dass sie in der gleichen Ebene liegen. Da sie mathematisch definiert sind, stören sie sich auch dort nicht, wo sie sich überlagern. Die Situation lässt sich programmtechnisch problemlos beschreiben. In den vom grafischen Programm erstellten Bildern zeigt das Überlagerungsdreieck (unteres Dreieck) aber ein fraktales Muster. Die Ursache dafür ist, dass der Rendering-Algorithmus die Lage jedes Punktes aus den

Wissenschaft und Technik, Gabriele Gramelsberger/Peter Bexte/Werner Kogge (Eds.), Transcript 2013, S. 171 – 192.

[37] Wie weit instrumentelle Vernunft und zweckrationales Wirtschaftsdenken führen können, zeigt das Beispiel der Fleischproduktion. Im Ziel der kostengünstigen und schnellen Produktion werden die Bedürfnisse einer artgerechten Haltung ausgeblendet.

[38] Im konkreten Beispiel wurde CINEMA 4D benutzt. In vielen 3D Programmen ist die beschriebene Aktion entweder untersagt oder sie wird in der Software abgefangen und 'repariert', indem eine der Überlagerungsflächen entfernt wird.

Eckpunkten des zugehörigen Dreiecks approximiert. Aufgrund der begrenzten Rechengenauigkeit erscheint mal die weiße und mal die schwarze Fläche vorne. Die fraktale Struktur der *Arithmetik endlicher Genauigkeit* führt zum gezeigten Bild, das wenn es interaktiv gesteuert wird, je nach Blickpunkt sein Muster wechselt.

Abbildung 1: Zusammenbruch der Euklidischen Idealisierung im konkreten Rechenprozess. Die Fraktale Struktur der *Arithmetik endlicher Genauigkeit* wird sichtbar.

Die Hardware-Begrenzung der Maschine zerstört hier sich im Prozess der Konkretisierung die idealisierte Geometrie. Unsere Lebenswelt kennt keine unendlich dünnen Dreiecke. Wenn wir zwei Dreiecke übereinander legen, wird immer eines oben liegen und das andere verdecken. Je nachdem, ob die Dreiecke opak oder transparent sind, werden wir die Farbe des oben liegenden Dreiecks oder eine subtraktive Mischfarbe sehen. Die unendlich dünnen Dreiecke der Euklidischen Geometrie formal im Rechner abzubilden birgt keinerlei Schwierigkeit, erst im Rechenprozess bricht die Idealisierung zusammen und die materiellen Bedingungen des Rechnens werden sichtbar. Eine ideale Maschine mit unendlicher Genauigkeit würde das Muster nicht erzeugen. Was wir sehen, ist die fraktale Struktur der Arithmetik endlicher Genauigkeit. Natürlich kann das '*Problem*' in der Software leicht behoben werden, aber das ist nicht der Punkt. Worauf es hier ankommt ist die Tatsache, dass im formalen Modell der Geometrie das Problem nicht existiert, erst die praktische Rechnung generiert das Artefakt. Es kommt hier nicht etwa ein Phänomen zurück, das wir im Prozess der Abstraktion weggenommen hätten, sondern etwas Neues das mit der Materialität des Rechnens zu tun hat.

Die Kontingenzen durch den Betrachter

Interfaces realisieren eingangsseitig Transformationen von komplexen offenen Umgebungen auf digitale Werte. Bei solchen Aufzeichnungsprozessen kommen oft Nebeneffekte ins Spiel. So enthält das von einer Kamera erzeugte Bild niemals nur die vom Kameramann intendierte Information. Umgekehrt generieren Ausgabeprozesse, die symbolische Repräsentationen in Bilder umsetzen, ebenfalls unvermeidlich Nebeneffekte. Bei komplexen Szenen wird das auf dem Interface erscheinende Bild immer auch Unbeabsichtigtes enthalten, das jenseits der bewussten Setzung durch den Programmierer oder Designer liegt. Winograd und Flores bezeichnen diese Phänomene auch als 'unbeabsichtigte Repräsentation'[39]. Beispielsweise kann der Betrachter Kreise auf dem Bildschirm sehen, obwohl weder ein explizites Konstruktionsprinzip für Kreise auf Codeebene implementiert ist, noch der Programmierer in irgendeiner Phase der Programmentwicklung an Kreise gedacht hat. Interfaces können eingangsseitig semantische Potentiale sowohl verringern, als auch ausgangsseitig neue semantische Potentiale erzeugen.

[39] Winograd und Flores, Erkenntnis Maschinen Verstehen, Rotbuch Verlag, Berlin 1989, S. 155.

Das Bewusstsein, dass jede Rezeption eines Kunstwerks die aktive Partizipation des Betrachters erfordert, ist eine kritische Erkenntnis zeitgenössischer Ästhetik. Kein Kunstwerk kann demnach einem Betrachter genau das mitteilen, was der Künstler beabsichtigt. Marcel Duchamp nennt dies den 'persönlichen Kunst-Koeffizienten', der das Verhältnis zwischen dem 'Unausgedrückten-aber-Beabsichtigten' und dem 'Unabsichtlich-Ausgedrückten' beschreibt. Auch Umberto Eco macht in seiner Poetik des offenen Kunstwerks deutlich, dass das Verhältnis zwischen Autor und Rezipient immer schon einen gewissen Grad an Co-Kreation voraussetzt. In jeder ästhetischen Erfahrung kommt dem Betrachter selbst eine konstitutive Rolle zu.[40] Dies scheint nun aber gerade den Zielen der mathematischen Formalisierung und dem Code-Denken entgegenzulaufen, wo ja gerade versucht wird, alle Mehrdeutigkeiten auszuschalten.

Donald Norman unterscheidet drei mentale Modelle, die er bei der Entwicklung jedes Artefakts im Spiel sieht: das Design-Modell, das User-Modell und das System-Image.[41] Das Design-Modell ist das Konzept, das der Entwickler entwirft und vor Augen hat, wenn er das Artefakt herstellt. Das User-Modell ist die Vorstellung, die der Benutzer sich vom Artefakt macht, während er mit ihm umgeht. Benutzer und Designer kommunizieren nur durch das Artefakt: seiner physischen Erscheinung, seinem Verhalten, der Art und Weise wie es reagiert und den Beschreibungen, die dem Benutzer über das Artefakt vorliegen. Der Designer wird also versuchen, seine Entscheidungen so zu treffen, dass das gebaute System und sein Konzept möglichst gut übereinstimmen und das Artefakt dem Nutzer die wesentlichen Merkmale seines Entwurfs kommuniziert und für die Interaktion offen legt. Bei Computeranwendungen stehen nun aber Codes im Zentrum des Zusammenspiels zwischen Designer und System. Das Hauptmerkmal von Programmen besteht gerade darin, dass sie jeden Interpretationsspielraum ausräumen und so ihr Verhalten prinzipiell möglichst eng an Intentionen des Entwurfs koppeln können. Der Code ist das, was wirklich ins System eingeschrieben ist, er ist die objektive Basis der Modelle, die sich Designer und Nutzer vom System machen; er ist also der Teil, über den sich nicht streiten lässt und der nicht der freien Interpretation offen steht. Im Umgang mit dem System werden sich beim Designer und beim User Vorstellungen aufbauen, die stark differieren und die vor allem auf Code-Ebene alleine nicht dingfest zu machen sind. Jenseits des objektiv im Code Eingeschriebenen gibt es vielfältige Zuschreibungen, die vor allem mit den im Interface gezeigten Inhalten zu tun haben und darauf abzielen, weitere bedeutende Aspekte der Computeranwendungen zu erfassen. Erst Einschreibungen und Zuschreibungen zusammen vervollständigen den Kanon der möglichen Sichtweisen. Im Gegensatz zum Usability-Ansatz Normans, geht es im künstlerischen Kontext gerade nicht darum, nur effiziente Handlung zu ermöglichen, sondern auch offene Handlungs- und Interpretationsfelder anzubieten. Unter Offenheit einer Computeranwendung soll dabei ganz allgemein die Eigenschaft verstanden werden, dass dem Benutzer bzw. Rezipienten jeweils neue Handlungs- und Interpretationsspielräume eröffnet und gewisse Flexibilitäten und Durchlässigkeiten für dessen Intentionen geschaffen werden. Es fehlen noch die Strategien, die es erlauben, zwischen den beiden Schichten zu vermitteln und die Spannungen zwischen Einschreibungen und Zuschreibungen kreativ zu nutzen.

[40] Vgl. Georg Trogemann, Jochen Viehoff, a.a.O., S. 146ff.
[41] Donald Norman, *The Psychology of Everyday Things*, Basic Books, 1988, S 189f.

«Wir müssen uns zu einer Auffassung über die Struktur der Dinge, deren Wandel und deren allgemeinste Größen entschließen, wie sehr sie auch durch die Erfahrung gewandelt werden mögen.»[42] Rupert Riedl

ANHANG A:
Die Dreiteilung der Maschine: Hardwarenetze – Interfaces – Codes

1982 wurde die Firma Sun Microsystems mit der Vision gegründet: «*Das Netz ist der Computer!*» Keine Prognose zur Zukunft des Computers dürfte sich in den zurückliegenden 25 Jahren als richtiger erwiesen haben. Wenn wir die wesentlichen Charakteristiken unserer heutigen programmierbaren Maschinen aufzählen, müssen wir neben der klassischen Trennung in Hard- und Software vor allem die Vielfalt der Interfaces und die vollständige Vernetzung nennen. Die programmierbare Maschine zerfällt nicht mehr wie in den Anfangszeiten ihrer Entwicklung in den Rechner und sein Programm, sondern in drei komplex zusammenspielende Komponenten: Rechnernetze, Interfaces und Software. Das Zusammenspiel von Material und Form, sowie die Spannung zwischen Ästhetik und Vernunft, sind in der Praxis wesentlich an Interfaces und Codes gekoppelt. Rechnernetze sind das notwendige Substrat auf dem sich das zeitliche Verhalten programmierter Objekte überhaupt erst realisieren kann.

Hardwarenetze sind Zusammenschlüsse aus verarbeitenden, speichernden und kommunizierenden Recheneinheiten. Die grundlegenden Funktionalitäten solcher vernetzter Systeme – das Speichern, Übertragen und Rechnen – finden heute millionenfach gleichzeitig und verteilt statt. Jeder Knoten dieses Netzwerks ist in der Lage Berechnungen auf der Basis von Codes auszuführen, Daten (und Codes) zu empfangen, zu speichern und zu versenden. Solche Netze verbinden nicht nur Personal Computer oder Handys weltweit miteinander, sondern auch unzählige Mikrocontroller, die – mit kleinsten Sensoren ausgestattet – in alle möglichen Alltagsgegenstände eingebaut sind und per Funk miteinander kommunizieren. Die physikalischen Eigenschaften der Rechner sind für die Zeitlichkeit der Medien verantwortlich. Die Algorithmen selbst haben keine Zeit, sie kennen nur das Aufeinanderfolgen einzelner Schritte. Erst die tatsächliche Ausführung der Schritte erzeugt das von der konkreten Hardware abhängige Zeitverhalten.[43] Die Hardware und die auf ihr ablaufenden Prozesse generieren erst mit Hilfe 'organisierter Materie' ein Zeitsignal und stellen es der Software zur Verfügung. Nur dadurch kennt die Software Zeit und kann Rechen- und Kommunikationsprozesse algorithmisch kontrollieren.

Interfaces sind jene Punkte der Rechnernetze, an denen die zeichenbasierten maschinellen Prozesse mit ihrer Umgebung in Verbindung treten. Digitale Rechner können prinzipiell nur Zeichen verarbeiten, d.h. symbolische Eingaben, die in der Regel für objektivierte Sachverhalte der Außenwelt stehen, in symbolische Ausgaben transformieren. Interfaces sind die Vermittlungseinheiten zwischen den symbolischen Prozessen in den Rechnernetzen und der nicht-symbolischen Umwelt. Mit Sensoren sind wir in der Lage, eingabeseitig Ereignisse automatisch aus der Umgebung zu extrahieren und als Zeichen zu codieren. Die innere Struktur des Interfaces – der Messprozess – legt dabei fest, welche Ereignisse der Umwelt das Innere des Systems erreichen. Durch den diskreten Messvorgang

[42] Rupert Riedl, Strukturen der Komplexität. Eine Morphologie des Erkennens und Erklärens, Springer Verlag, Berlin Heidelberg 2000, S. 102.

[43] Unter dem Stichwort „Deutschland jubelt zeitversetzt" war während der Fußballweltmeisterschaft 2006 ein Beispiel für die Materialität der Rechnernetze zu erleben, das mehrere Millionen Menschen irritierte. Kabelzuschauer sehen das Live-Signal einer Fußballübertragung ca. 4 Sekunden früher als DVB-T Empfänger, für die Dynamik eines Fußballspiels bedeuten diese 4 Sekunden Welten. So kam es, dass die verkabelten Nachbarn bereits Tore bejubelten, während auf dem eigenen DVB-T Fernseher noch nichts auf das Ereignis hindeutete.

im Sensor findet somit eine Objektivierung der Umwelt statt die alle Unschärfen eliminiert. Durch das Interface-Mapping legt das System aber gleichzeitig seine eigene Blindheit gegenüber allen anderen, nicht zur Messung ausgewählten Phänomenen fest. Ausgabeseitig findet in den Interfaces der umgekehrte Prozess statt. Zeichen werden zu Materialprozessen und Handlungen. Das rationale, text- und zahlenbasierte Denken verschiebt sich damit zurück auf die Ebene der sozio-kulturellen Praxis von Tönen, Bildern, Bewegungen usw.

Die in den Rechnernetzen prozessierten ***Codes*** haben ihre Wurzeln einerseits in der Mathematik, andererseits in der Elektrotechnik. Computercodes werden deshalb als technik-wissenschaftliche Hybridobjekte aufgefasst und einerseits als mathematisch-formale Beschreibungen interpretiert, andererseits wird ihnen – im Zusammenspiel mit dem Substrat eines Rechners – die Kraft, zu operieren und selbständig zu handeln, zugesprochen.[44] Diese Codes sind nun aber nicht mehr wie zu Beginn der Rechnerentwicklung an eine einzelne Recheneinheit gebunden, sondern können sich mehr oder weniger frei durch die Rechnernetze bewegen. Die Hardware wird damit zur wählbaren Umgebung. Die Zeichen flottieren durch die Netze, werden archiviert, kopiert und zur Ausführung gebracht. Für den Benutzer von Computern ist es meist nicht mehr transparent, wo und wann seine Anforderungen bedient werden und auf welche Weise dies im Detail geschieht. Die verteilt interagierenden Betriebssysteme – ebenfalls komplexe Codesysteme – verwalten vollkommen eigenständig die Ressourcen der Rechnernetze, teilen Laufzeiten zu, erstellen Kopien, konvertieren Daten von einem Format in ein anderes, versuchen das System gegen unerwünschte Zugriffe und Aktionen zu schützen und vieles mehr. Die maschinenseitige Betriebssystem und die Anwendersoftware verbinden sich zu einer komplexen Aufführung.

Hardware und Interfaces sind aufgrund ihrer Materialität 'naturgebunden', sie unterliegen den Gesetzen der physikalischen Welt, d.h. sie verbrauchen Energie, um ihre Arbeit zu leisten, unterliegen Alterungsprozessen und sind anfällig für allerlei Defekte. Die Codes dagegen sind direkt mit dem menschlichen Denken und der menschlichen Sprache verbunden, sie sind der algorithmischen Rationalität verpflichtet. Die Begrenzungen der Codes unterliegen nicht mehr wie die Hardware direkt den Gesetzen der materiellen Welt, ihre Begrenzungen sind die Grenzen unseres Denkens und unserer Vorstellungskraft[45]. Nicht die Digitalität computerbasierter Medien ist wesentlich, sondern ihre sprachliche Basis. Da wir auf dieser sprachlichen Grundlage auch stetige Modelle beschreiben und behandeln können, ist die Digitalität der Maschine sekundär. Eventuelle Einschränkungen folgen aus der mangelnden Mächtigkeit der Programmiersprache oder – was der übliche Fall sein dürfte – der Komplexität der Aufgabe. Fehler, die im Zusammenhang mit Codes auftreten sind deshalb immer konzeptionelle Fehler und nie 'Materialfehler' im physikalischen Sinn. Das heißt, Code-Fehler entstehen dann, wenn die Vorstellungen, die sich der Programmierer vom seinem Programm macht nicht mit den Funktionen übereinstimmen, die es tatsächlich berechnet.

[44] Heike Stach, *Programme, zwischen Notation und Organismus – Zur kulturellen Konstruktion des Computer-Programms*, Deutscher Universitäts-Verlag, Wiesbaden 2001.

[45] Abgesehen von grundsätzlichen Berechenbarkeitsgrenzen, die in der Praxis aber eher eine untergeordnete Rolle spielen. Außer, wenn mit Hilfe von Programmen semantische Aussagen über andere Programme gemacht werden sollen.

ANHANG B:
Die Dreiteilung der Maschine: Hardwarenetze – Interfaces – Codes

```
int x1 = 70, x2 = 130, x3 = 210, x4 = 330;
int y1 = 50, y2 = 130, y3 = 250;
int X =380, Y = 300;

void setup() {
  size(X, Y);
  background(255);

  fill(255, 0, 0); rect(x1, y1, x2-x1, y2-y1);
  fill(0, 0, 255); rect(x3, y1, x4-x3, y2-y1);
  fill(255, 255, 0); rect(x1, y3-40, x2-x1, 40);

  strokeWeight(6);
  line(x1, 0, x1, X);
  strokeWeight(4);
  line(x2, 0, x2, X);
  line(x3, 0, x3, X);
  strokeWeight(6);
  line(x4, 0, x4, X);
  line(x1, y3-40, x2-2, y3-40);

  line(0, y1, X, y1); line(0, y2, X, y2); line(0, y3, X, y3);
  save("MyImage.jpeg");
}
```

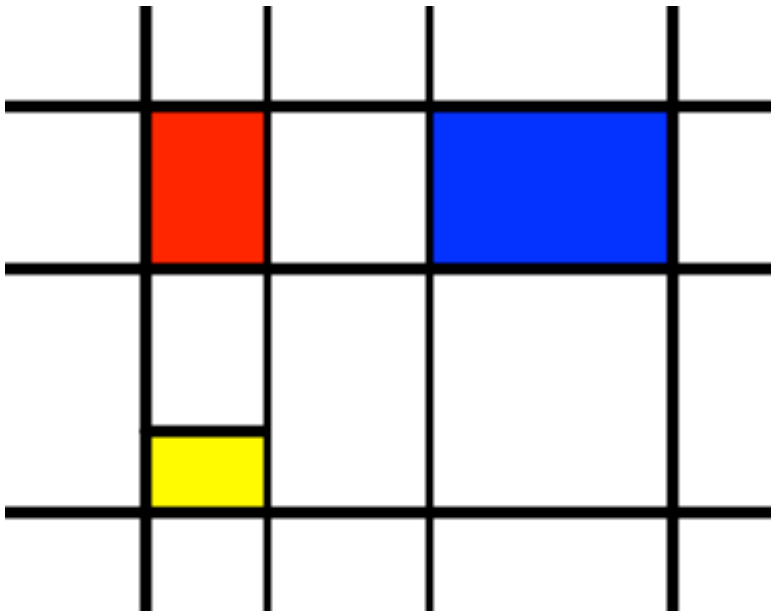


Abbildung 2: Trivialbeispiel einer programmierten Geometrie. Dargestellt sind der *Processing*-Code und das resultierende (Mondrian affizierte) Bild. Neben den eigentlichen geometrischen Objekten (Linienverlauf in der Ebene, Lage der Rechtecke) werden auch schon Materialeigenschaften im Code festgelegt (Hintergrund, Farben, Liniendicken).

> *«Der Künstler als Programmierer schafft Werke als Klassen von Werken. Er denkt grundsätzlich, wenn er schafft. Er denkt an alle Bilder, die ein inneres Band verbindet. Er denkt an Bilder als Klassen, die es berechenbar zu realisieren gilt.»*[46] Frieder Nake

In der Terminologie Frieder Nakes sind hier *Unterfläche* und *Oberfläche* eines digitalen Bildes dargestellt.[47] Gemäß der Konventionen der Programmiersprache *Processing* verläuft die X-Achse des kartesischen Koordinatensystems von links nach rechts, die Y-Achse von oben nach unten. Der Nullpunkt des Koordinatensystems liegt folglich in der oberen linken Bildecke. Die Variablen x1 – x4 markieren Start- und Endpunkte auf der X-Achse, die Variablen y1 – y3 entsprechend Start- und Endpunkte auf der Y-Achse. Die Variablen X und Y legen im Befehl `size(380,300)` die Bildgröße auf 380x300 Pixel fest. Alles was außerhalb dieses Bereiches gezeichnet wird, ist unsichtbar. In *Processing* bedeutet die Syntax `line(x1,y1,x2,y2)`, dass eine Linie vom Startpunkt (x1,y1) zum Endpunkt (x2,y2) gezogen wird. Der Befehl `rect(x1,y1,x2,y2)` zeichnet entsprechend ein Rechteck mit linker oberer Ecke im Punkt (x1,y1) und rechter unterer Ecke im Punkt (x2,y2). Der einfache Bildaufbau, d.h. die Lage der Linien und Rechtecke kann damit bereits nachvollzogen werden. Da wir nicht feste Werte für die Punkte und damit den Verlauf der Linien verwenden, sondern Variablen, realisiert der angeschriebene Algorithmus nicht nur ein Bild, sondern eine ganze Klasse von Bildern. Unter Beibehaltung des Verfahrens können bei Veränderung der Integer-Werte für die x- und y-Variablen unzählige Bilder generiert werden. Die Vorstellung, die wir mit einem Wertepaar (x1,y1) verbinden entspricht exakt der abstrakten Idee des Euklidischen Punktes in der Ebene. Abstrakte Idee und mathematischer Formalismus stehen hier vollkommen im Einklang. Zahlenwerte, die zum Beispiel auf dem Zahlenstrahl visualisiert sind, besitzen in unserer idealisierten Vorstellung ebenfalls keine Ausdehnung. Sobald die abstrakten Ideen von Punkten, Linien und Flächen akzeptiert sind, ist auch die Syntax des hier dargestellten Programms leicht zu verstehen. Wir können mit den Punkten in der geometrischen Ebene rechnen wie mit anderen arithmetischen Größen auch.

Die restlichen Befehle des Programms beziehen sich bereits auf Materialeigenschaften des Bildes. Mit `strokeWeight()` wird die Pinselstärke und mit `fill()` die aktive Farbe festgelegt. Der Programmierer definiert hier bereits materielle Parameter des resultierenden Bildes. Interessant ist, dass auch die Materialeigenschaften auf dieser Ebene noch formal beschrieben werden, also mit Hilfe von Zahlenwerten, die ihrer sinnlichen Qualitäten beraubt sind. Die Fülle des Konkreten kehrt erst durch der Materialität der Codeausführung zurück. Mit der materiellen Anreicherung kommt auch der Schmutz der dreidimensionalen Welt zurück, Rechengenauigkeiten sind begrenzt und Materialfehler die Normalität. Farbwerte verwandeln sich in echte Farben und Größenangaben von Objekten sind nicht mehr nur Zahlen, sondern belegen realen Raum. Code und Interface müssen hierfür aufeinander abgestimmt sein. In unserem trivialen Beispiel wurde mit der Ausführung des Programms für jeden Pixel des 380x300 Punkte großen Bildes ein Farbwert berechnet und das aus RGB-Zahlenkolonnen bestehende digitale Bild im jpeg-Format abgespeichert. Das Bild als farbige Oberfläche entsteht aber erst in einem weiteren Transformationsprozess. Die Realisationsfläche kann ein Computerbildschirm sein, ein projiziertes Beamer-Bild, ein Blatt Papier, das von einem Laserdrucker bedruckt wird, oder eine Leinwand, auf die ein Farbplotter Farbe aufträgt. Jede dieser Schnittstellen prägt dem Bild seine eigenen materiellen Qualitäten auf. Für den Betrachter resultieren daraus neue Interpretationsspielräume, Wahrnehmungsqualitäten und Kontingenzen. Natürlich steuert der Code den Prozess, die Fülle des physischen Resultats enthält er aber nicht.

---

[46] Frieder Nake, space.color, in: *Algorithmik – Kunst – Semiotik. Hommage für Frieder Nake*, Synchron Wissenschaftsverlag 2003, S. 139.

[47] Frieder Nake, Das Doppelte Bild, in: Horst Bredekamp, Matthias Bruhn, Gabriele Werner (Herausgeber), *Bildwelten des Wissens*, Akademie-Verlag 2006, S. 40-50.

ANHANG C:
Die Goldbachsche Vermutung

In der Zahlentheorie gibt es unbewiesene Probleme, die auf einfache Weise algorithmisch formuliert werden können. Die Goldbachsche Vermutung wurde im Jahre 1742 vom Mathematiker Christian Goldbach in einem Brief an den Leonhard Euler als Vermutung formuliert. Die Goldbachsche Vermutung lautet wie folgt:

Jede gerade Zahl größer als 2 kann als Summe zweier Primzahlen geschrieben werden.

(4 = 2 + 2)
6 = 3 + 3
8 = 3 + 5
10 = 3 + 7 = 5 + 5
12 = 5 + 7
14 = 3 + 11 = 7 + 7
16 = 3 + 13 = 5 + 11
18 = 5 + 13 = 7 + 11
20 = 3 + 17 = 7 + 13
22 = 3 + 19 = 5 + 17 = 11 + 11
24 = 5 + 19 = 7 + 17 = 11 + 13
...

Tabelle 1: Beweis der Goldbachschen Vermutung für die ersten 10 geraden Zahlen.

Für die ersten Zahlen sind die Ergebnisse in Tabelle 1 aufgelistet. Es lässt sich leicht ein Programm skizzieren (Abbildung 3), das genau dann anhält, wenn die Goldbachsche Vermutung falsche ist. In dieser Formulierung bedeutet die Richtigkeit der Goldbachschen Vermutung, dass unser Programm nie halten wird. Obwohl sich eine Reihe namhafter Zahlentheoretiker mit der Goldbachschen Vermutung befasst haben, konnte sie bis heute weder bewiesen noch widerlegt werden. Es kann also niemand sagen, ob unser kleines Programm je halten würde oder nicht.

```
n = 4;
conjecture = true;
while (conjecture == true) {
   n = n + 2;
   conjecture = false;
   for (i = 3, to n-2, n+2) {
        j = n - i;
        if (i prim && j prim){
            conjecture = true;
            break;
        }
   }
}
print("Goldbach's conjecture is false");
```

Abbildung 3: Programmskizze für die Berechnung der Goldbach-Zerlegung.