Code

Eine elementare Einführung in die Programmierung als künstlerische Praktik

Der Ausgangspunkt für dieses Interview ist ein Buch, das Sie 2005 gemeinsam geschrieben haben: «CodeArt Eine elementare Einführung in die Programmierung als künstlerische Praktik» (Trogemann/Viehoff 2005). Das sehr umfang- und facettenreiche Buch ist – meines Wissens – das einzige, welches sich in diesem Ausmaß der Programmierung im künstlerischen Kontext widmet. Der erste Teil befasst sich mit der programmierbaren Maschine sowie einer umfassenden Auslegung der damit einhergehenden Fragestellungen und schafft eine Verbindung zum Kontext der künstlerischen und kulturellen Praxis.

Art

Georg Trogemann und Jochen Viehoff
Interview, durchgeführt von Adriana Mikolaskova Nautsch

Der zweite Teil besteht aus einer für Anfängerinnen und Anfänger sehr gut nachvollziehbaren, sorgfältig konzipierten Programmiereinführung, welcher auch das sogenannte «Codekit» – eine Sammlung von Beispielen und Modulen – angegliedert ist. Im Gegensatz zu allgemeinen Programmiereinführungen beziehen sich die Beispiele auf Bilder, Animationen und Klänge und ermöglichen so relativ rasch einen Zugang zu entsprechenden Experimenten. Die Einführung wird ergänzt durch sehr interessante und anregende Exkurse der Kultur- und Technikgeschichte und bettet die technischen Aspekte in philosophische Betrachtungen ein.

Während Sie, Herr Trogemann, als Professor für Experimentelle Informatik an der Kunsthochschule in Köln tätig sind, zum Thema schreiben und forschen, leiten Sie, Herr Viehoff, das größte Computermuseum der Welt, das Heinz Nixdorf MuseumsForum, und fotografieren. Wie sind Sie zu diesem Gebiet gekommen? Wie ist Ihr Interesse an der Verknüpfung von Programmierung und Kunst entstanden?

Georg Trogemann (GT): Als Informatiker haben mich Algorithmen und Programmierung natürlich seit meinem Studium beschäftigt. Dabei hat mich immer gewundert, wie technisch reduziert und wie wenig kulturgeschichtlich und disziplinübergreifend Programmierung behandelt wurde. Ich habe mich deshalb sehr früh auf die Suche nach den Wurzeln unseres Tuns in der Informatik gemacht.

Das Buch war dann das Ergebnis eines im Jahr 2000 gestarteten und auf fünf Jahre angelegten Modellversuchsprogramms des Bundes mit dem Namen «Kulturelle Bildung im Medienzeitalter», an dem mein Kollege Hans Ulrich Reck und ich für die Kunsthochschule für Medien Köln (mit einem von 23 Modellversuchen aus 13 Bundesländern) teilgenommen hatten. Ziel des Programms war, innovative Modelle für den kreativen und kompetenten Umgang mit neuen Medientechnologien in der kulturellen Bildung und Ausbildung zu entwickeln und zu erproben und dabei insbesondere ästhetische Erfahrungen einzubeziehen.

Durch meine Professur für Experimentelle Informatik an der Kunsthochschule für Medien Köln, der Verankerung meines Fachs im Bereich Kunst- und Medienwissenschaften sowie der täglichen Praxis mit jungen Künstlerinnen und Künstlern war der Versuch, Algorithmik mit Kunst und Kultur zusammenzubringen, natürlich naheliegend. Mein Hauptinteresse bei diesem Vorhaben war, die damals weitgehend unsichtbare Seite

der Medienkunst – die Programmierung – sichtbar werden zu lassen, aber nicht als bloße Technik, sondern in ihrem kulturellen Reichtum und ihren vielfältigen historischen Bezügen.

Jochen Viehoff (JV): Schon während meines Physikstudiums und der sich anschließenden Promotion in der theoretischen Physik mit einem Schwerpunkt auf Computersimulationen und der Programmierung paralleler Supercomputer hat mich die Kunst und insbesondere die Fotografie bereits sehr interessiert. An der Kunsthochschule für Medien in Köln bot sich für mich dann die perfekte Möglichkeit, beide großen Interessensgebiete spannend miteinander zu verknüpfen. Nach sechs Jahren in Köln bin ich dann erst als wissenschaftlicher Kurator ans Heinz Nixdorf MuseumsForum (HNF) gewechselt, ein wunderbares Computermuseum, welches ich seit 2013 leite. Auch wenn die Zeit nun knapper geworden ist, fotografiere ich immer noch für das Tanztheater Wuppertal Pina Bausch. Aber in der Kombination von Computern und Kunst sehe ich nach wie vor großes Potential, selbst wenn viele experimentelle Arbeiten aus dem Bereich der sogenannten Medienkunst heute als fertige ‹Produkte› käuflich zu erwerben sind.

In welche Richtung hat sich Ihr Interesse seit dem Erscheinen des Buches weiterentwickelt?

GT: Tatsächlich gibt das Buch meine Sichtweise auf Algorithmen und Programmierung bis zum Jahr 2005 wieder. Seit der Fertigstellung des Buches ist nicht nur die Bedeutung der Algorithmen für alle Bereiche der Gesellschaft sehr viel klarer hervorgetreten, sondern auch meine eigenen Formen der Betrachtung und Behandlung des Themas haben sich seitdem weiterentwickelt. Meine Auseinandersetzung mit Algorithmen ist mittlerweile in einen größeren Kontext eingebunden,

«Backtracking», aus: Viehoff, Jochen: Die algorithmische Welt, Bildband mit Fotografien, welche mit Szenen und Motiven aus Alltag und Natur die Prinzipien der Programmierung illustrieren.

beispielsweise der Frage, warum es unserer Kultur nie gelungen ist, den Bereich der Technik, zu dem auch die Programmierung und Digitalisierung zählen, geistig wirklich in den Griff zu bekommen. Nach wie vor stehen sich die von C. P. Snow[1] beschriebenen zwei Kulturen, die geisteswissenschaftlich-literarische einerseits und die naturwissenschaftlich-technische andererseits, so diametral gegenüber, dass eine Verständigung bis heute kaum möglich scheint. Die Probleme, die dadurch bei der Bewältigung der Folgen unseres technischen Handels entstehen, sind inzwischen unübersehbar. Gerade vor diesem Hintergrund ist es wichtig, immer wieder Brücken zu schlagen und gerade auch Verbindungen zwischen Technik und Kunst zu erproben.

JV: Für mich ist die Thematik noch viel stärker in den Mittelpunkt gerückt, weil wir mittlerweile davon ausgehen sollten, dass Programmierung als Kulturtechnik nicht nur für Künstlerinnen und Künstler absolut relevant ist, sondern genau genommen zur schulischen Grundausbildung in einer vernetzten digitalen Wissensgesellschaft gehören sollte. Die Ausrichtung ist ja die gleiche: mit spannenden Inhalten und dem technikhistorischem Kontext unserer digitalen Maschinen die Neugierde der jungen Menschen zu wecken. Das ist auch eine der Hauptaufgaben des HNF. Projekte wie «You can code» («Jeder kann programmieren»), aber auch die *Do-it-yourself*-(DIY) und die *Maker-Bewegung*[2] geben die Richtung vor. Hier muss die Bildungspolitik nachziehen.

Wie war damals die Resonanz auf dieses Buch?
Wissen Sie, wer es gelesen hat und wie damit gearbeitet wurde?
GT: Das Buch war in erster Linie an Medienkünstlerinnen und -künstler sowie Gestalterinnen und Gestalter adressiert. In die-

sem Feld wurde es auch wahrgenommen und kommentiert. Ich bekomme aber bis heute immer wieder auch interessante Rückmeldungen aus dem Bereich der Geisteswissenschaften, der Architektur sowie von Programmiererinnen und Programmierern aus anderen Disziplinen, für die algorithmische Fragestellungen und die Grundlagen der Digitalisierung im eigenen Fach ebenfalls immer bedeutsamer werden.

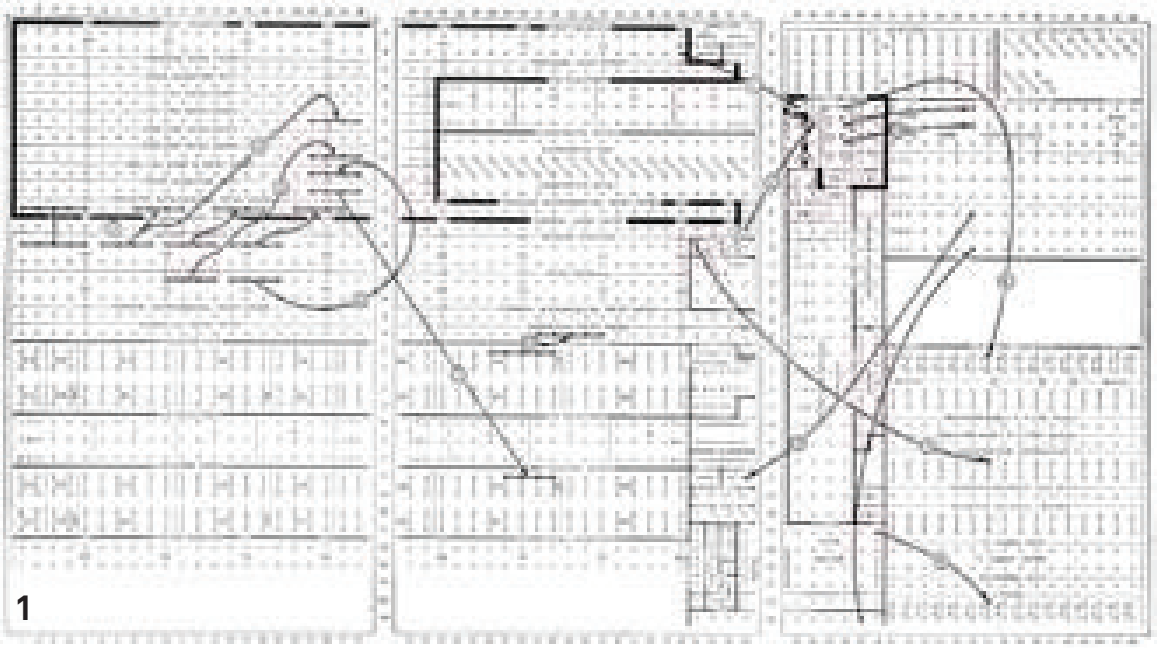
Das Buch lässt sich auf zwei grundsätzlich verschiedene Weisen rezipieren. Man kann sich damit begnügen, die ersten 150 Seiten als allgemeine Einführung ins Thema durchzuarbeiten und sich die anschließenden Programmierteile oberflächlich ansehen bzw. in Zwischenspielen lesen. Man kann sich umgekehrt aber auch auf die Programmierteile konzentrieren und das Buch vor allem als praktische Einführung in die Programmierung verstehen. Jedenfalls wurde offensichtlich auf diese beiden unterschiedlichen Weisen damit gearbeitet, wobei man sagen muss, dass das Buch zwölf Jahre alt ist und mittlerweile andere praktische Programmiereinführungen zur Verfügung stehen, die aktueller und stärker auf heute relevante Bereiche zugeschnitten sind. Seit dem Erscheinen des Buchs haben sich insbesondere im künstlerischen Kontext verschiedene Werkzeuge durchgesetzt, die in der Summe die Bedürfnisse von Kunstschaffenden gut abdecken, etwa *Processing, Arduino, vvvv* und einige andere Entwicklungsumgebungen, insbesondere auch für den Soundbereich. Hierzu gibt es jeweils auch hervorragende Einführungen.

Inwiefern hält das Interesse an – auf welche Aspekte werden Sie auch nach längerer Zeit angesprochen?

GT: Was sich seit Erscheinen des Buches natürlich nicht verändert hat, ist der kultur- und technikgeschichtliche Hintergrund der Programmierung. Wobei es ja nicht um eine präzise

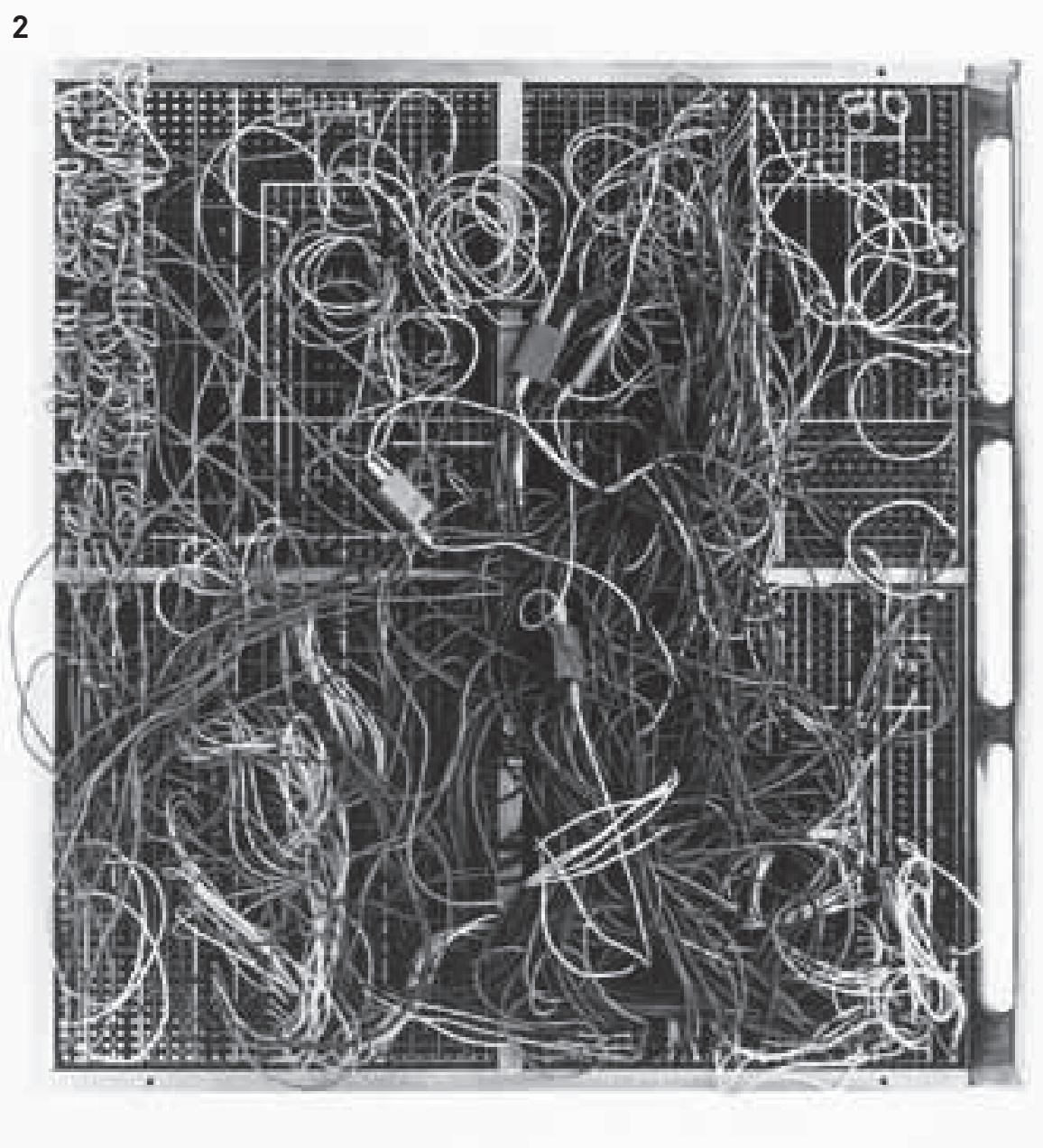
«Sequentiell und parallel» aus Viehoff, Jochen: Die algorithmische Welt, Bildband mit Fotografien, welche mit Szenen und Motiven aus Alltag und Natur die Prinzipien der Programmierung illustrieren.


APARTNER



1

2



1. IBM 402/403 Steckfeld-Diagramm (Plugboard) mit eingezeichneten Verbindungen, aus dem Kapitel «Die Struktur der programmierbaren Maschine» (Trogemann/Viehoff 2005, S. 78), mit freundlicher Genehmigung der IBM-Archive.

2. IBM 407 Steckfeld, mit freundlicher Genehmigung der IBM-Archive.

und lückenlose geschichtliche Aufarbeitung der Programmierung geht, sondern vielmehr um den Versuch, die geistigen Wurzeln der Programmierung offenzulegen sowie die Algorithmik vor allem in der Geistesgeschichte zu verorten und als spannendes Feld menschlichen Handlungsdenkens darzustellen, jenseits spezieller Problemlösungen und konkreten Codings. Hier hat das Buch anscheinend bis heute ein Alleinstellungsmerkmal.

JV: Ich würde hier nicht von anhaltendem Interesse sprechen, sondern eher von einer Renaissance der in *CodeArt* enthaltenen Ideen und Inhalte. Wie gesagt, der Ansatz «Jeder kann Programmieren» in Kombination mit preiswerten Einplatinencomputern – wie der britische *Micro4Bit* oder das deutsche *Calliope-Projekt* (https://calliope.cc) – zeigen, welchen Stellenwert Programmierung in der schulischen Laufbahn haben sollte.

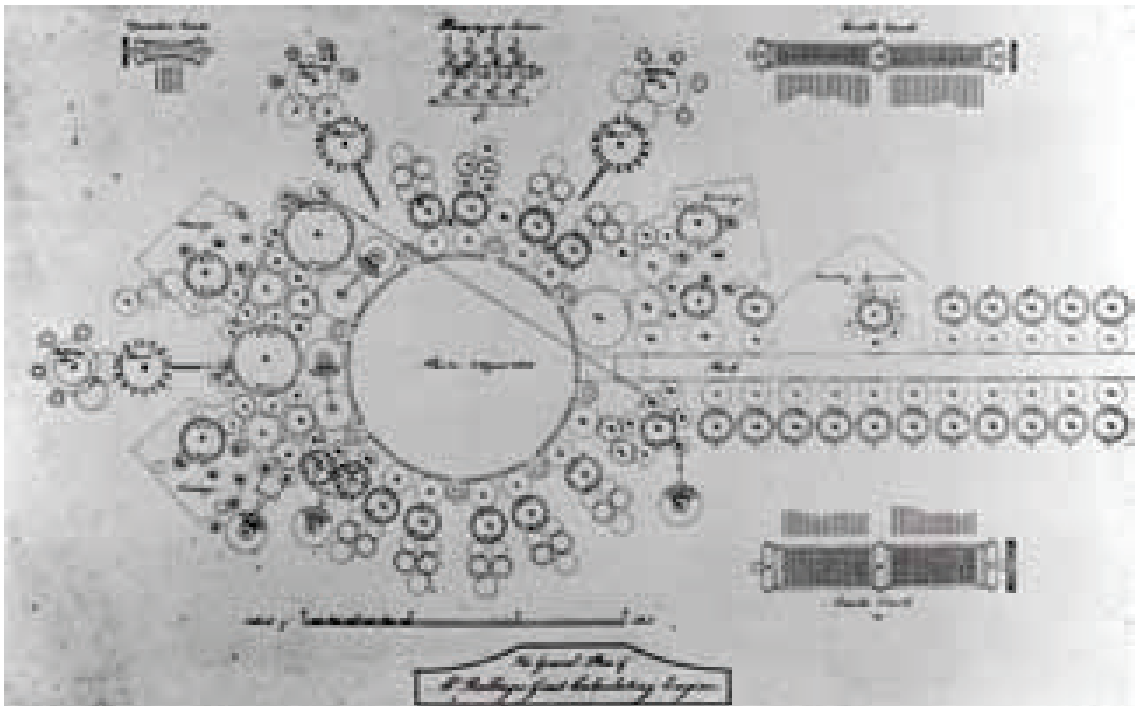
In den letzten Jahren hat das Programmieren unter dem Namen «Generative Gestaltung» in der visuellen Gestaltung, im Design

und der Architektur Einzug gehalten. Creative Coding, das Programmieren, in der Absicht, etwas Ausdrucksstarkes – statt Funktionales (bzw. Administratives) – zu erzeugen, wäre ein weiteres Schlagwort … Inwiefern gibt es dabei Berührungspunkte zu Ihren Absichten? Oder ist das etwas ganz anderes?
GT: Generative Gestaltung und Creative Coding sind natürlich wichtige Teilaspekte der Programmierung, nicht nur im künstlerischen Umfeld. Hierdurch lassen sich mindestens zwei Aspekte klar zeigen, die für das gesamte Feld der sogenannten Digitalisierung Relevanz besitzen.

Zum Ersten tritt in der Generativen Gestaltung und auch beim Creativen Coding die Zweiteilung computergenerierter Phänomene, die Frieder Nake die *Oberfläche* und die *Unterfläche* der Bilder nennt, deutlich hervor. In *CodeArt* (Trogemann/Viehoff 2005) habe ich das als «Janusköpfigkeit des Pixels» bezeichnet, ohne zu diesem Zeitpunkt Nakes Arbeiten hierzu zu kennen. Unterfläche und Oberfläche charakterisieren aber nicht nur algorithmische Bilder, sondern alle computergenerierten Artefakte. Man kann den algorithmischen Phänomenen nicht gerecht werden, wenn man nur die semiotisch-rationale Ebene der Programme betrachtet, also die Zeichen der Programmierebene und deren Logik. Der Programmtext – die vom Computer bearbeitbare Unterfläche – repräsentiert nur die funktionalen, logischen und handlungsorientierten Aspekte, die Oberfläche dagegen – die phänomenologische Ebene – wird erst wahrnehmbar (sichtbar, hörbar und als Handlung wirksam), wenn das Programm läuft. An der Oberfläche erscheinen dann die eigentlichen Phänomene, d. h. die für den Menschen wahrnehmbaren Wirkungen des Programms. Eingriffe durch die Nutzerin oder den Nutzer sind aber auf beiden Ebenen möglich, auf der Oberfläche nennen wir das dann *Interface Design,* auf der Unterfläche *Programmie-*

rung. Oberfläche und Unterfläche sind während des Programmablaufs untrennbar miteinander verbunden – wie zwei Seiten einer Medaille oder wie die zwei Gesichter von Janus. Beide Seiten sprechen unterschiedliche menschliche Kompetenzen an, etwa Vernunft, Logik und analytische Zerlegung auf der einen, Geschmack, Ästhetik und eine synthetisierende Verbindung auf der anderen Seite.

Zum Zweiten verdeutlichen Generative Gestaltung und Creative Coding das ungeheure fiktionale Potential von Algorithmen. *Algorithmen spannen Handlungsräume auf; die Programmiererin oder der Programmierer muss stets einen Möglichkeitsraum denken.* Welche konkreten Pfade eines Programms wirklich realisiert werden, zeigt sich erst, wenn das Programm läuft. Algorithmen sind somit immer extreme Verdichtungen von Möglichkeitsräumen. Es geht darum, deutlich zu machen, dass wir nicht nur Aspekte unserer Welt abbilden, sondern Fiktionen realisieren. Diesen Möglichkeitsraum, den die/der Programmierende durch den Code festlegt, kann sie/er selbst in der Regel nicht vollständig überblicken. Die Korrektheit komplexer Software kann deshalb bis heute nur durch Softwaretests für sorgfältig ausgewählte Testdaten sichergestellt werden. Bei evolutionären Algorithmen, in der genetischen Programmierung, aber auch in Computerspielen werden diese künstlichen Handlungsräume explizit genutzt, um zu Lösungen zu kommen, die auch die Person, die programmiert, überraschen. Die Entfaltung konkreter Realisierungen kann dabei einerseits durch Interaktionen mit den Nutzerinnen und Nutzern gesteuert werden, andererseits aber auch durch das Programm selbst, indem Exploration und funktionale Bewertung sich abwechseln. *Für die Programmiererin oder den Programmierer geht es dabei immer um Abwägungen zwischen Freiheit und Kontrolle. Wo kann ich auf Kontrolle zugunsten von Überraschungsmomenten verzichten?*



Wo muss die Kontrolle gewährleistet bleiben? Wer Buchhaltungssoftware für eine Bank schreibt oder Software für die Steuerung eines Flugzeugs, wird an solchen Freiheitsgraden von Software kaum interessiert sein. Beim kreativen Umgang mit Algorithmen steht das Unvorhersehbare, der Missbrauch, das Gegen-den-Strich-Bürsten von Erwartungen dagegen oft im Zentrum.

In einer aktualisierten Fassung von *CodeArt* müssten diese Aspekte sicher stärker herausgearbeitet werden. Meiner Ansicht nach werden sie bisher in der Literatur noch nicht wirklich tief genug reflektiert. Jedenfalls nicht ausreichend, um im Kontext der Geistesgeschichte unserer Technologien begreifbar und so mit anderen kulturellen Bereichen, in denen Fiktion eine bedeutsame Rolle spielt, verbindbar zu werden.

JV: Ich würde es so formulieren: Die Informatik hat viel zu lange ihre Lerninhalte an die Mathematik und naturwissenschaftliche oder technische Inhalte angelehnt. Programmieren war halt etwas für Nerds und mathematisch-technisch

Schema der Analytical Engine von Charles Babbage, aus dem Kapitel «Universelle Turingmaschinen & minimale Kontrollstrukturen» (Trogemann/Viehoff 2005, S. 247).

begabte Menschen. Aber über die Inhalte aus ganz anderen Themenbereichen wie Kunst, Gestaltung, Games oder Sound erwächst neues Interesse daran, Dinge selber umzusetzen, als Programm, als Hardware oder gleich als Kombination der beiden Welten. *Sicherlich ist Programmieren keine leicht zu erlernende Kulturtechnik. Aber das würde ja auch niemand beispielsweise über die Mathematik oder das Erlernen einer Schriftsprache sagen.* Trotzdem gehören sie zur schulischen Grundausbildung. Und je früher man damit anfängt, desto einfacher ist es für junge Menschen. Viele Projekte erproben heute Lerninhalte der Informatik bereits an Grundschulen. Hier ist es ganz wichtig, das Interesse und Neugierde zu wecken, insbesondere auch bei Mädchen.

In CodeArt verwenden Sie die – immer noch – weit verbreitete und eingesetzte Programmiersprache Java. Wie beurteilen Sie die neu entstandenen Frameworks wie Processing, Pure Data, Nodebox etc.?

GT: Hier haben natürlich wichtige Entwicklungen stattgefunden. Eine aktuelle Version von *CodeArt* würde vermutlich auf *Processing* oder *Arduino* basieren oder sich vielleicht auch gar nicht auf eine bestimmte Plattform festlegen, sondern gerade zeigen, dass sie sich alle miteinander verbinden lassen, dass sie sich gegenseitig ergänzen und unterschiedliche Stärken und Schwächen haben. Auch die Entwicklungen rund um den *Raspberry Pi* sind hier wichtig. Es geht bei *CodeArt* aber weniger um eine konkrete Programmiersprache oder eine bestimmte Plattform, es geht um eine bestimmte Art des Denkens, die in einer ganz bestimmten kulturellen Tradition steht. *Der algorithmische Zugriff auf die Welt ist viel älter als Computer und Programmierung.* Aber erst durch die explizite Fassung dieses handlungsorientierten Denkens als maschinell prozessierbarer Code konnte die Algorithmik ihre wirkliche Macht entfalten. Jahrhunderte lang wurden Algorithmen im Kopf mit Papier und Bleistift prozessiert. Die Auslagerung in die Maschine hat nun das Potential, bisherige Prozesse komplett umzukrempeln.

JV: Ich würde heute nicht noch einmal den vollen Funktionsumfang einer so mächtigen Programmiersprache wie Java für das Projekt wählen. Es gibt mit *Processing* und all den anderen Plattformen viel besser geeignete Frameworks. Aber der Trend geht eindeutig – aus meiner Sicht – dahin, Plattformen zu wählen, die ganz unmittelbar zwischen der Hard- und Software sitzen, wie die Entwicklungsumgebungen für *Arduinos* oder andere Einplatinencomputer, bei denen sich Coding und Löten eigener Hardware sinnvoll ergänzen. Grafische Programmierumgebungen wie *Scratch* bieten hier insbesondere jungen Menschen Einstiegsmöglichkeiten in die Programmierung mit sehr niedriger Hemmschwelle.

Während der Untertitel von CodeArt «Programmierung als künstlerische Praktik», lautet, findet mit «Code und Material», dem Untertitel Ihres späteren Buches – Exkursionen ins Undingliche – eine leichte Verlagerung statt.

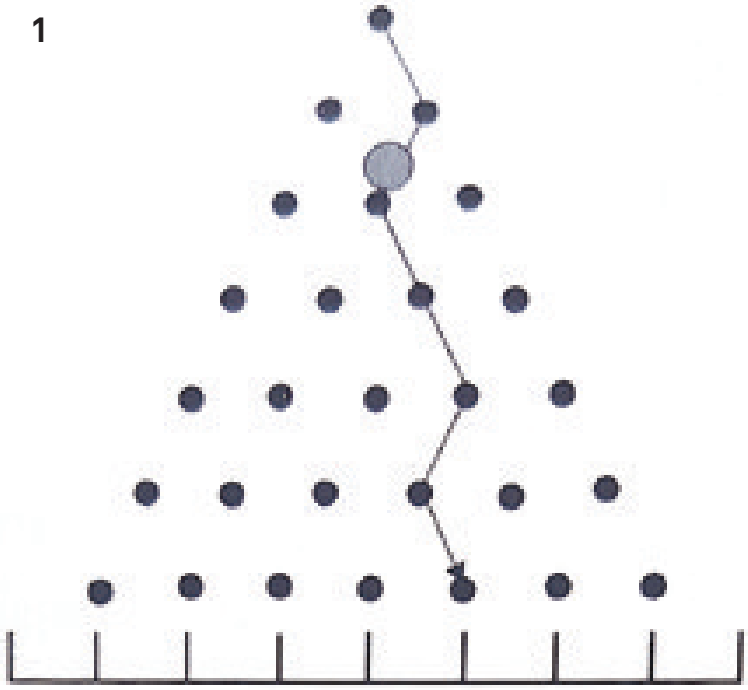
GT: *Es geht vor allem darum, zu zeigen, dass Programme nicht auf den Computer begrenzbare Artefakte sind. Software ist immer in die Wirklichkeit eingebettet und wird in ihrem spezifischen Umfeld real wirksam.* Es gibt dabei zwei grundlegende Formen der Einbettung in die Wirklichkeit: Das ist zum einen Software als Medium in menschlichen Kommunikationsprozessen. Die Wirksamkeit solcher Systeme können wir derzeit in den sogenannten sozialen Netzen beobachten. Zum anderen handelt es sich um die Einbettung in Materialprozesse, also alle Formen der Herstellung und Kontrolle materieller Prozesse und Artefakte. Schon der Begriff der Information, der ja angeblich grundlegend für unsere ganze Epoche sein soll, vereint kommunikative und materielle Aspekte. Etwas muss sich einerseits ins Material einprägen, um zur Information werden zu können, andererseits muss dieser materielle Abdruck aber auch innerhalb eines Kommunikationsprozesses verstanden werden. «Code und Material» zielt auf die vielfältigen neuartigen Verbindungen ab, die algorithmische Codes mit Materialien und Materialprozessen eingehen können. Hier öffnet sich ein vollkommen neues Handlungsfeld. Die verstärkte Hinwendung zum Material innerhalb von Programmierkursen an der Kunsthochschule für Medien war aber auch eine Reaktion auf sich wandelnde künstlerische Interessen. Nach dem ‹Immaterialisierungsstreben› der ersten Digitalisierungswelle hat das analoge Material in der Kunst eine erstaunliche Renaissance erlebt. Es kann aber nicht darum gehen, das Material gegen den Code auszuspielen, stattdessen sollte vielmehr ihre Verbindung untersucht werden. Inzwischen ist ohnehin in vielen

1. Die «pin machine» (auch Galtonbrett) nach Sir Francis Galton (1877), aus dem Kapitel «Zufall und Determinismus» (Trogemann/Viehoff 2005, S. 353), die den Effekt der ‹natürlichen Auslese› zeigen soll. Auf einem Brett sind in regelmässigen Abständen Nägel eingeschlagen, die in aufeinanderfolgenden Reihen jeweils um den halben Abstand der Nägel versetzt sind. Lässt man von oben eine Anzahl von Kugeln durch das Nagelbrett fallen, dann soll eine Kugel, die auf einen Nagel trifft, mit gleicher Wahrscheinlichkeit nach links oder nach rechts abgelenkt werden und danach auf einen Nagel der darunterliegenden Nagelreihe treffen. Dort wiederholt sich der Vorgang, bis alle Nagelreihen durchlaufen sind und die Kugel schliesslich in einem bestimmten Fach zum Liegen kommt. Wenn wir eine hinreichend grosse Zahl von Kugeln durch das Nagelbrett fallen lassen, zeigt sich die typische Normalverteilung der Kugeln auf die Fächer.
2. Visualisierung der «pin machine» (Galtonbrett) von Sir Francis Galton, aus dem Kapitel «Zufall und Determinismus», erschienen in: «Typical Laws of Heredity III», Nature Magazine, vom 19. April 1877.

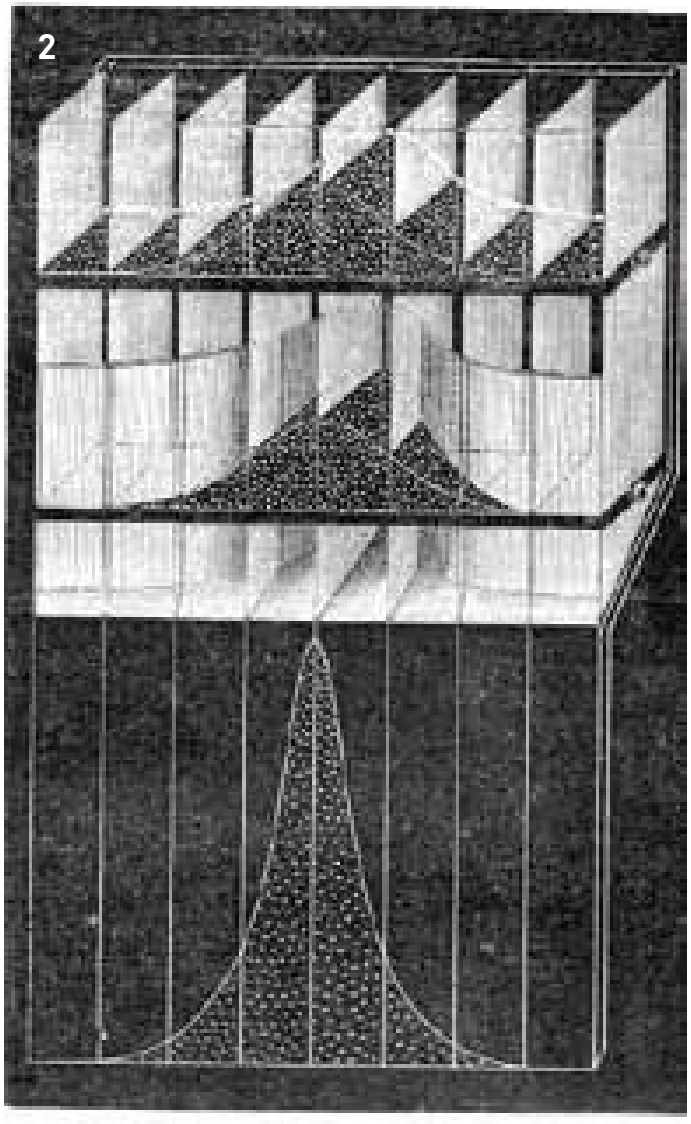
Feldern, nicht nur in der Architektur und der Gestaltung deutlich geworden, dass die Digitalisierung eine enorme Bedeutung für alle Formen der materiellen Produktion hat. Die neuen Handlungsräume, die sich aus der Verbindung von Code und Material ergeben, sind aber natürlich auch für Künstlerinnen und Künstler ein interessantes ästhetisches Explorationsfeld.

Während Sie in CodeArt den Studierenden Programmierbeispiele zur Verfügung stellen, beschreiben Sie im oben erwähnten Buch das «lab3», in welchem mit Microcontrollern, Sensoren und Motoren gearbeitet wurde. Auch für diese Arbeit haben Sie eine Umgebung zusammengestellt, die aber auch eigens für das Lab konzipierte Hardware beinhaltet. Wie kommt es zu dieser Verschiebung? Welches sind/waren Schnittstellen zur sogenannten Robotik?

1



2



GT: Mit dem Erfolg der Physical-Computing-Plattform *Arduino* haben sich zwangsweise auch unsere Programmierkurse für Studierende verändert. Wir sind schon sehr früh in die Programmierung mit Arduino eingestiegen und haben dann 2009 den *Sensor-Aktor-Shield* für unser Lab entwickelt. Das ist eine aufsteckbare Hardware-Extension für das Arduino Board. Wir hatten festgestellt, dass bestimmte Nutzungen des Boards sich wiederholen, zum Beispiel die Ansteuerung von Motoren, die höhere Spannungsversorgungen brauchen als das Arduino Board sie liefert, auch dass es für Testzwecke beim Physical-Computing hilfreich ist, ein Potentiometer zu haben und verschiedene Leuchtdioden, man häufig analoge Ein-/Ausgänge benötigt und so weiter. All das haben wir auf dem Sensor-Aktor-Shield zur Verfügung gestellt und so die Hürde für den Programmiereinstieg unter Arduino noch einmal deutlich reduzieren können. Damit war es auch für Personen, die mit Programmieren anfangen, möglich, innerhalb eines zweiwöchigen Kurses eine interessante kinetische Skulptur, eine kleine interaktive Arbeit oder eine einfache Robotik-Anwendung zu realisieren. Aber auch diese Entwicklung wurde inzwischen von Arduino und anderen Anbietern überholt. Solche Erweiterungen für das Arduino Board gibt es inzwischen für wenige Euros zu kaufen.

Inwiefern greift die (technisch orientierte) Maker-Szene mit all ihren Plattformen und FabLabs[3] Ihre Vorstellungen der Verbindung von rationaler und ästhetischer Tradition auf? Oder was fehlt dazu?

GT: *FabLabs* greifen tatsächlich etwas Wichtiges auf und entwickeln es für die Materialbearbeitung weiter, nämlich den Grundgedanken, dass wir auch in Zeiten autonomer Fabriken und komplexer werdender Technologien selbst etwas tun kön-

nen, dass das Machen, die Fähigkeit, etwas selbst herzustellen oder auch nur reparieren zu können, einen Wert an sich beinhaltet. Hinzu kommt die Erfahrung, dass die Sprache, das Schreiben und die zwischenmenschliche Kommunikation andere Widerstände besitzen als die Arbeit mit Material. Das ‹Selbstmachen› kann nicht nur helfen, souveräner gegenüber der Technik zu werden, sondern eine ganz andere Art der Verbindung mit der Welt herstellen und sinnstiftend für das eigene Leben sein. Die Bestseller *Shop Class as Soulcraft* von Matthew B. Crawford (2009) oder auch das schon etwas ältere Buch des vor wenigen Monaten verstorbenen Robert M. Pirsig *Zen and the Art of Motorcycle Maintenance* (1974) behandeln genau diese Fragen in hervorragender Weise. Bei FabLabs vermisse ich allerdings genau diese Ebene der Reflektion, also die Verbindung des Machens mit dem tieferen philosophischen Nachdenken über das Machen.

FabLabs greifen aber auch einen anderen Gedanken auf, der ursprünglich aus der Softwareentwicklung kommt, und entwickeln ihn weiter, nämlich die *FLOSS-Initiative (Free/Libre Open Source Software Initiative).* Generell sind Makerspaces oder auch Repair Cafés deshalb natürlich sehr zu unterstützen.

Welche Künstlerinnen und Künstler finden Sie im Zusammenhang mit der Programmierung interessant und nachvollziehbar – auch für ein jüngeres, jugendliches Publikum, Lernende an Gymnasien?

JV: Am spannendsten finde ich derzeit künstlerische Arbeiten, die die Bereiche Künstliche Intelligenz und die Programmierung von tiefen neuronalen Netzwerken mit der Robotik, also den technischen Maschinen, zusammenbringen. Arbeiten, in denen die lernfähigen Algorithmen eine reale körperliche Erscheinung einnehmen. Das ist an Schulen natürlich nur

schwer einzubinden. Hier sehe ich das größere Potential bei der DIY- und Maker-Szene, wenn es um den gezielten Einsatz in Schulen geht.

In Ihrer fotografischen Arbeit, Herr Viehoff, machen Sie Prinzipien der Programmierung in Bildern aus dem Alltag sichtbar.[4] Sie betreiben quasi ein Reengineering der Realitätskonstruktion, die beim Konzipieren von Programmiersprachen stattfindet … Wie kam es zu dieser Arbeit?

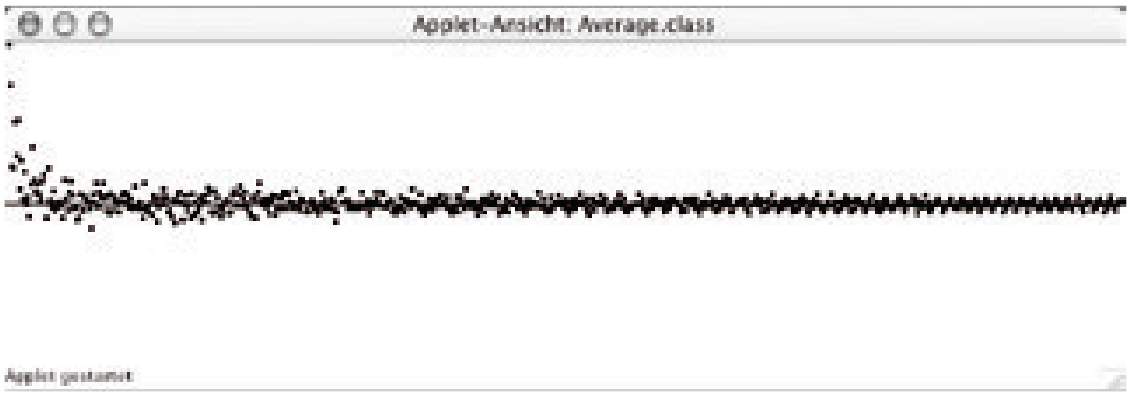
JV: Tatsächlich hat mich mein Vorgänger, der ehemalige Museumsdirektor Norbert Ryska, mit einem Arbeitsauftrag auf das Thema angesetzt: «Fotografieren Sie doch mal Software!». Ich habe dann angefangen, mir Gedanken zu machen, welche Bilder denn dabei entstehen könnten, wenn die Hardware, die Geräte und ihre Peripherie, keine Rolle spielen sollten. Nach ersten zaghaften Versuchen bin ich zwangsläufig bei der Idee des Algorithmus gelandet. *Software schafft Möglichkeiten, damit bestimmte Prozesse ablaufen können, aber unser Alltag ist ebenfalls geprägt von unzähligen Strukturen, die den Ablauf bestimmter Prozesse ermöglichen, dabei Spuren hinterlassen. Überhaupt hat sich die Informatik in der Umsetzung von Algorithmen in Software sehr viel von natürlichen Prozessen abgeschaut.* Diesen Zusammenhang aufzuzeigen, das war mein Anliegen in der Fotoarbeit «Die algorithmische Welt – Prinzipien der Programmierung im Alltag». Die Arbeit ist übrigens auch in unserem Museum in der Abteilung «Software» ausgestellt.

Verstehe ich Sie insofern richtig, als dass auf einem ähnlichen, umgekehrten Weg aus der Auseinandersetzung mit der Programmierung ein allgemeiner Erkenntnisgewinn resultiert – über unsere Welt, über Ordnungsprinzipien, Handlungsstrukturen und -muster?

JV: Einfache Antwort: Ja, ganz unbedingt. Sonst hätte ich mich nicht so lange und intensiv im Rahmen des Fotoprojektes damit auseinandergesetzt.

In Ihrem Buch CodeArt schreiben Sie von der Programmierung als elementarer Kulturtechnik, Herr Viehoff erwähnt dies auch in einigen seiner Antworten. Würden Sie, Herr Trogemann, die Programmierung auch immer noch so bezeichnen? Diese Sichtweise wird in letzter Zeit wieder vermehrt aus verschiedenen Richtungen und Gründen kritisiert … Es geht bei den Argumenten um die Vereinnahmung der Bildung durch die Wirtschaft, um die Nutzlosigkeit von Programmierungskenntnissen angesichts selbstlernender Systeme etc.

GT: Nun, wenn Software-Systeme heute Flugzeuge, Kernkraftwerke, Fabriken und bald auch autonome Autos kontrollieren. Wenn Aktienverkäufe durch Algorithmen ganze Finanzmärkte zusammenbrechen lassen. Wenn Cyberkriege, also Kriegsführungen auf der Basis von Informationstechnik, stattfinden und Wahlen durch Manipulationen sozialer Netze beeinflussbar sind. Wenn nicht zuletzt selbst der Zugang zum Wissen durch algorithmische Strukturen kontrolliert wird, etwa durch den Page-Rank-Algorithmus von Google, und wir unsere zukünftigen Lebenspartnerinnen oder -partner auf Vermittlungsplattformen algorithmisch auswählen lassen, glauben Sie dann nicht auch, dass es sich vielleicht lohnen könnte, einen Blick darauf zu werfen, auf welcher Grundlage das alles geschieht? Gerade, um sich nicht durch die Wirtschaft vereinnahmen zu lassen, sondern um diskursfähig zu werden. Um eine Basis zu haben, auf der wir all das, was da geschieht, überhaupt beurteilen zu können. Auch der Hinweis auf selbstlernende Systeme hilft wenig, da diese natürlich auch programmiert sind. Die Lernstrategien sind algorithmische Ver-



1

2

1. Annäherung des Mittelwertes an den theoretischen Wert der Verteilung, aus dem Codekit, Kapitel «Gute und schlechte Zufallszahlen» (Trogemann/Viehoff 2005, S. 362).

2. Direkte Korrelationen in aufeinanderfolgenden Zufallszahlen kann man im Falle schlechter Parameter (rechts) an der Struktur in der Verteilung erkennen. Gute Zufallszahlen füllen die Fläche gleichmässig aus (links); aus dem Codekit, Kapitel «Gute und schlechte Zufallszahlen» (Trogemann/Viehoff 2005, S. 363).

fahren, die Frage wird hier also nur um eine Ebene verschoben. Wie sich all das entwickeln wird, kann natürlich niemand für einen längeren Zeitraum voraussagen. Niemand weiß, wie Programmiersprachen in fünfzig Jahren aussehen, ob ‹Software› dann immer noch als Aneinanderreihung elementarer Operationen realisiert wird. Vielleicht sind die Computer, die es dann zu kaufen gibt, keine Universalmaschinen mehr, nicht frei programmierbar, sondern intransparent und funktional geschlossen. Was man aber sicher sagen kann, ist, dass die Art, wie wir unsere Welt objektivieren und abstrahieren, sie verfügbar und zuhanden machen, nicht verschwunden sein wird. Wie wir das bewerkstelligen, das kann man mithilfe von Programmierung, die auf sich selbst reflektiert, verstehen lernen. Man begreift dann vor allem auch, dass Vorstellung und Ergebnis oft auseinandergehen, dass Nebenwirkungen unserer Handlungen nie ganz auszuschließen sind, selbst in so extrem

engen Umgebungen wie denen der Programmierung. Das Problem mit Programmierkursen, nicht nur an Gymnasien, scheint mir eher darin zu liegen, dass es derzeit nur wenige Personen gibt, die solche Kurse, die sich weit jenseits bloßer Programmier-Skills bewegen, wirklich anbieten können. Hier muss vielleicht erst die nächste Generation heranwachsen.

Um auf die Frage von vorhin zurückzukommen, also: *Ja,* es handelt sich bei der Programmierung um eine Kulturtechnik. *Dieses gesamte Feld komplett den Technikern zu überlassen, ist nicht nur gefährlich, man beschneidet sich selbst und ohne Not auch neuer ästhetischer Handlungsspielräume.*

Was denken Sie über das Verhältnis der Programmiermöglichkeiten, die Kunstschaffenden sowie Gestalterinnen und Gestaltern zur Verfügung stehen, und den uns umgebenden immer komplexeren Softwaresystemen?

GT: Programme, die Künstlerinnen und Künstler, Gestalterinnen und Gestalter schreiben, unterliegen anderen Anforderungen als die Programme professioneller Software-Entwicklerinnen und -entwickler, die viel weitreichendere Qualitätsstandards erfüllen müssen. Künstlerische Software dient oft einem konkreten individuellen Projekt, sie ist oft fehlerbehaftet und schlecht kommentiert. Nach Abschluss des Projektes wandert das Programm in den Mülleimer oder bestimmte Teile werden in neuen Projekten weiterverwendet. *Die Programmierungen sind oft auch weniger komplex als viele kommerzielle Anwendungen, dennoch können programmierende Künstlerinnen und Künstler aufgrund ihrer Programmiererfahrung sehr gut einschätzen, was Software prinzipiell leisten kann und wie die Systeme grundsätzlich funktionieren. Das ist enorm wichtig, um kritikfähig zu sein.*

Als wichtigen Aspekt der Programmierung führen Sie die Realitätskonstruktion und die damit verbundenen Formalisierungs- und Abstraktionsprozesse auf. Auch im Kunstunterricht – in allen Medien – spielen diese eine große Rolle. Der Kunstunterricht setzt Lern- und Erkenntnisprozesse in Gang, die denjenigen der Formalisierungs- und Abstraktionsprozessen programmierter Arbeiten nicht unähnlich sind. Inwiefern sehen Sie eine Verbindung zwischen den im Falle des Programmierens und den im Fach Kunst thematisierten und erfahrbar gemachten Prozessen?
GT: Ich glaube ja, dass wir durch das Programmieren weniger etwas über die Welt an sich erfahren können als vielmehr über die Denkmuster, mit denen wir uns Welt zuhanden machen. Nicht Erkenntnis steht hier im Vordergrund, sondern *Poiesis,* also das Machen, Konstruieren, Herstellen. Damit Aspekte unserer Welt zu von der Maschine behandelbaren Zeichen werden können, müssen sie in Bezug auf bestimmte Blickwinkel der Betrachtung von Unwesentlichem und Zweideutigkeiten gereinigt werden. *Formalisieren bedeutet immer, von realen Verhältnissen zu abstrahieren, zurückzutreten, zu de-kontextualisieren und auf ein gezieltes Interesse hin Abgrenzungen und Reduktionen vorzunehmen.* Auf diesem Weg in die Maschine wird die Fülle der Phänomene auf in sich inhaltslose Zeichen reduziert. Bedeutung tragen nur die Relationen zwischen den Zeichen. Beim Ablauf des Programms sind diese Zeichenrelationen nun aber aktiv und erzeugen ihrerseits neue wahrnehmbare Phänomene. *In den prozessierenden Programmen läuft der Abstraktionsprozess der Programmierung quasi rückwärts ab, als Konkretisierung und Re-Kontextualiserung.* Die Materialität der Maschine (die Interfaces) laden die nun sichtbaren Bilder, hörbaren Klänge und wirksamen Handlungen mit Mehrdeutigkeiten und Unschärfen auf. Die Phänomene, die dabei erzeugt werden, sind aber nicht identisch mit denen, die in der Abs-

traktionssphase weggeworfen wurden. Es entsteht Unbeabsichtigtes, Nebeneffekte, was neue Interpretationsspielräume eröffnet. All das sind auch Strategien der Kunst.

Was sich hier abspielt, ist generell von sehr allgemeiner Bedeutung für unseren handlungsorientierten Umgang mit der Welt. In der Philosophie ist vor allem die Abstraktionsseite gut behandelt worden: die Umkehrung der Abstraktion, die Konkretisierungsseite, d. h. die Aufladung von Zeichenprozessen mit Phänomenen und Mehrdeutigkeiten, ihre Re-Materialisierung ist allerdings computerspezifisch und erfordert ganz neue eigene Betrachtungen und Untersuchungen. Die Re-Kontextualisierung, die dabei auch stattfindet, ist dagegen wieder eine bekannte künstlerische Strategie. *Insbesondere die Frage, wie Dinge sich zeigen, wie sie in Erscheinung treten und sich manifestieren, wie sie in einem bestimmten Kontext wirken, das dürfte für den Kunst- und Medienunterricht bzw. für den Unterricht an Gymnasien generell von Bedeutung sein.*

Macht es aus Ihrer Sicht Sinn, die Programmierung auf dieser – gymnasialen – Stufe, im gestalterisch- künstlerischen Kontext zu thematisieren?

JV: Ja, von mir aus kann gerne auch im Kunstunterricht programmiert werden. Mein Anspruch aber, Programmierung als elementare Kulturtechnik in den gesamten Lehrplan an Schulen zu integrieren, geht hier viel weiter.

GT: Die Frage ist aus meiner Sicht weniger, ob Programmierung im Kunstunterricht an sich sinnvoll ist, sondern vielmehr, wie es sinnvoll gemacht werden kann. Als reine technische Disziplin oder als Methode der Bilderzeugung hat Programmierung wenig Berechtigung, weder im Kunstunterricht noch in sonst einem anderen Fach am Gymnasium.

Programmierung verstanden als poietisches Handeln, als Grundprinzip, wie wir Welt rationalisieren, abstrahieren, de-kontextualisieren und operationalisieren, wie wir ästhetische Artefakte erzeugen, das kann dagegen auch für den Kunstunterrricht sehr fruchtbar sein. Das erfordert ein ganz anderes Herangehen an die Programmierung, als es in Standardkursen vermittelt wird. Die Frage ist also nicht: «Macht es Sinn?», sondern «Haben wir Lehrkräfte, die das sinnvoll machen können?»

Aber diese Frage, wie auch schon einige vorangegangene, weisen meiner Ansicht nach auf ein sehr viel größeres gesellschaftliches Problem hin: Wir haben unsere akademische Welt sehr sorgfältig in eine geisteswissenschaftlich-literarisch-künstlerische und eine naturwissenschaftlich-technische getrennt. Diese Trennung erfolgt nicht erst an den Universitäten, sondern wird zumindest in Deutschland schon im Gymnasium durch die Einrichtung humanistischer und naturwissenschaftlicher Zweige vollzogen. Technik und Kunst lassen sich vor diesem Hintergrund nur sehr schwer zusammenbringen. Die Trennung in zwei Kulturen ist über Jahrhunderte eingeübt worden und lässt sich vermutlich auch in den Köpfen von Lehrenden nicht so leicht erschüttern. Unter anderen gesellschaftlichen Paradigmen würden sich solche Fragen aber gar nicht erst stellen.

Bei Aristoteles gab es beispielsweise noch drei Grundformen des Denkens: die Theorie, die Praxis und die Poiesis. Während die ersten beiden in den letzten zweitausend Jahren Karriere gemacht haben, wurde die Poiesis weitgehend vergessen. Die theoretische Vernunft betrachtet und analysiert. Sie bildet damit u. a. die Grundlage für die Erkenntnistheorie, die Logik und die Wissenschaftstheorie. Die praktische Vernunft entwirft die Regeln unseres Verhaltens. Zu ihr zählen u. a. die Moralphilosophie, die Ethik und die Politik. Die Poiesis dagegen stellt die Frage, wie das Neue in die Welt kommt. Sie

umfasst alles Tun, das Werke hervorbringt, die wenn sie erst hervorgebracht sind, für sich selbst bestehen und ihre Wirkung in der Welt entfalten. Die Arbeiten zur poietischen Vernunft sind leider bis heute fragmentarisch geblieben, was Georg Picht als eines der großen Verhängnisse der europäischen Geistesgeschichte identifiziert. Eines der gut ausgearbeiteten Fragmente einer Grundlegung der Poiesis ist die Ästhetik. Ästhetische Praxis und Programmierung gehören in dieser Betrachtungsweise von vorne herein zusammen und müssen nicht erst gerechtfertigt werden. *Erst vor dem Hintergrund einer Gesamtschau der poietischen Vernunft werden auch der offene Horizont unserer Welt und der vehemente Eingriff in den Zukunftsraum durch unser technisches Handeln sichtbar. Nicht nur die Gefahren und Möglichkeiten der Technik werden auf der Basis eines poietischen Verständnisses sichtbar, sondern auch die gemeinsamen Wurzeln von Poiesis, Poesie und Ästhetik.*

Es würde aber mindestens eine Generation dauern, um die Lehrkräfte auszubilden, die einen derartigen grenzüberschreitenden Unterricht an breiter gymnasialer Front machen können – von der Konsensfähigkeit des Vorschlags und dem notwendigen politischen Willen solches durchzusetzen ganz zu schweigen. Deshalb wird es vermutlich auch auf längere Sicht an einzelnen fähigen Lehrern hängen, die in der Lage sind, Programmierung kompetent, spielerisch und reflektiert zu unterrichten und gleichzeitig vielfältige kulturelle Querbezüge zu gesellschaftlichen, künstlerischen, erkenntnistheoretischen, ethischen und anderen Fragen herstellen können, sodass es Schülerinnen und Schülern einerseits Spaß macht, sich mit dieser teilweise sehr abstrakten Materie der Codes auseinanderzusetzen, sie andererseits aber auch allgemeinere Einsichten erlangen, wie menschliches handlungsorientiertes Denken funktioniert.

Dr. Jochen Viehoff leitet das Heinz Nixdorf MuseumsForum (http://www.hnf.de).

Prof. Dr. Georg Trogemann forscht und lehrt an der Kunsthochschule für Medien Köln (https://www.khm.de; http://georgtrogemann.de/).

Literatur

Trogemann, Georg/Viehoff, Jochen: CodeArt. Eine elementare Einführung in die Programmierung als künstlerische Praktik, Wien/New York (Springer) 2005.

Anmerkungen

1 Zu C. P. Snow und der Begrifflichkeit der Zwei Kulturen vgl. die folgenden Wikipedia-Beiträge, verfügbar unter: https://de.wikipedia.org/wiki/C._P._Snow und https://de.wikipedia.org/wiki/Zwei_Kulturen, abgerufen am: 21.11.2017.

2 Die Maker-Szene ist eine Subkultur, die man auch als Do-It-Yourself-Kultur mit dem Einsatz aktueller Technik beschreiben kann.

3 Ein FabLab (engl. fabrication laboratory – Fabrikationslabor), manchmal auch offene Werkstatt oder MakerSpace, ist eine offene, demokratische Werkstatt mit dem Ziel, Privatpersonen den Zugang zu Produktionsmitteln und modernen industriellen Produktionsverfahren (3D-Drucker, Laser-Cutter, CNC-Maschinen etc.) für Einzelstücke zu ermöglichen.

4 Vgl. Beitrag zur Fotoausstellung von Jochen Viehoff im Rahmen der Paderborner Fototage vom 06.06.–05.09.2010: «Die algorithmische Welt – Ablauf, Struktur und Kontrolle in Natur und Alltag», verfügbar unter: http://www.hnf.de/ausstellungen/rueckblick/die-algorithmische-welt.html, abgerufen am: 19.10.2017.