Generative und interaktive Computerkunst mit dem Quellcode als Medium

Christian Schulze

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Digital Arts

in Hagenberg

im Oktober 2012

© Copyright 2012 Christian Schulze

Diese Arbeit wird unter den Bedingungen der *Creative Commons Lizenz Namensnennung–NichtKommerziell–KeineBearbeitung Österreich* (CC BY-NC-ND) veröffentlicht – siehe http://creativecommons.org/licenses/by-nc-nd/3.0/at/.

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 8. Oktober 2012

Christian Schulze

Inhaltsverzeichnis

Vorwort

Ich möchte mich bei allen bedanken, die während meiner Schreibphase die häufigen Anmerkungen, was nicht alles generativ sei, ausgehalten und mich in dieser Phase unterstützt, motiviert, aufgemuntert und unterhalten haben.

Allen voran danke ich Sandra, die meine Hochs und Tiefs wohl am deutlichsten gespürt, aber immer toleriert und mit Fassung getragen hat. Die Gespräche mit ihr haben mir stets neue Ideen sowie Motivation gegeben und daher bin ich besonders dankbar für ihre Unterstützung.

Auch Tom sei gedankt. Er ist der beste Roomie, mit dem ich in meinem Leben ein Zimmer teilen durfte und auf den man zu jeder Zeit zählen kann, wenn z. B. das Auto mal nicht anspringt. Es war eine schöne Zeit, die ich immer in guter Erinnerung behalten werde.

Ebenso möchte ich meiner Familie, insbesondere meinen Eltern danken, die so viel für mich opfern, dass sie u. a. dafür über 1.000 km bei maximal 80 km/h auf sich nehmen, um ebendieses Auto wieder zu reparieren. Auch ihre finanzielle Unterstützung soll dankend erwähnt sein, ohne die ich kein so entspanntes und erkenntnisreiches Studentenleben hätte führen können.

Ein Dank geht auch an meinen Projektpartner Mo, ohne den unser Projekt nur halb so gut, halb so lang und halb so schnell fertig geworden wäre. Ich bin sehr zufrieden mit dem Ergebnis und er hat den entscheidenden Teil dazu beigetragen.

Des Weiteren danke ich Roland Keil, mein Thesis- und Projektbetreuer, der mich zu vielen Inhalten der Master Thesis und gestalterischen Aspekten des Projekts inspiriert hat. Außerdem hat sein Anspruch unser Projektteam immer wieder gefordert, aber seine Gelassenheit uns davor bewahrt, überfordert zu sein.

Mein Dank geht ebenso auch an meine Kommilitonen und Dozenten, die mich in Hagenberg sehr herzlich aufgenommen und unterstützt haben. Die Atmosphäre im Studiengang ist sehr unkompliziert und persönlich, nicht vergleichbar mit der Anonymität an vielen Universitäten und es freut mich, ein Teil dieser DA-Familie sein.

Kurzfassung

Generative und interaktive Kunst sind keine Kunstformen im eigentlichen Sinn. Sie erstrecken sich über viele Bereiche und bedienen sich verschiedenster Techniken und Werkzeuge. Der Computer hat sich dabei zum wichtigsten Werkzeug der generativen und interaktiven Künstler entwickelt. Die Kommunikation zwischen Künstler und Maschine findet einerseits über Applikationen statt, andererseits aber auch zunehmend direkt über vom Künstler verfassten Quelltext. Gestalter werden zu Programmierern und nutzen die Rationalität der Maschine, um kreative Prozesse auszuführen.

Der Quellcode wird dadurch zum Werkzeug des programmierenden Künstlers. Er steht in der Produktion als vermittelnde Instanz zwischen Künstler und Werk sowie zwischen Werk und Rezipient in der Präsentation. Auf dieser Basis entwickelt sich eine Auseinandersetzung mit der These, der Quellcode sei das zentrale Medium zwischen Künstler, Werk und Rezipient.

Die Ausführungen dieser Master Thesis setzen sich mit kreativen Prozessen in Form von generativen und interaktiven Kunstwerken auseinander, die in erster Linie mit der Hilfe von Computern produziert wurden. Dennoch wird zur Einführung auch die historische Entwicklung wiedergegeben, die ebenso Werke enthält, welche ohne Computer funktionieren bzw. produziert wurden. Des Weiteren enthält die Arbeit Erläuterungen zu dem Begriff des Zufalls und dessen spezielle Bedeutung in der Computerkunst.

Erläuterungen zu Techniken und Erscheinungsformen generativer sowie interaktiver Kunstwerke bilden die Grundlage zur Analyse des im Zuge dieser Arbeit durchgeführten Projekts Ars Rata. Das Kurzfilmprojekt wird in einer Werkanalyse vorgestellt und auf generative Merkmale hin untersucht. Die Ergebnisse der Auseinandersetzung und ein Ausblick auf die Bedeutung generativer und interaktiver Kunst in der Zukunft schließen die Master Thesis ab.

Abstract

Generative and interactive art are no art forms as such. They spread over many fields and make use of different tools and techniques. In the process the Computer has developed to the primary tool of the generative and interactive artists. The communication between artist and machine happens through applications on the one hand, but also directly through source code written by the artist. Designers become programmers and make use of the rationality of the machine to execute creative processes.

Thereby the source code becomes the tool of the programming artist. While in production it lies as mediating entity between artist and artwork as well as it lies between artwork and recipient during presentation. On this basis the master thesis deals with the assumption, that the source code is the central medium between artist, artwork and recipient.

The master thesis deals with creative processes in terms of generative and interactive artworks, that were produced with the aid of computers in the first place. However the historical development will be presented as introduction, which contains also artworks, that were produced without computers. Furthermore the master thesis contains explanatory notes on the use of random functions and their special meaning concerning computer art.

Remarks about techniques and forms of appearance of generative and interactive artworks provide the basis for the analysis of the project Ars Rata, which was realized as part of the thesis. The short film project will be presented in a separate chapter where it will be analysed for its generative characteristics. The results of the analysis and prospects of the future relevance of generative and interactive art will complete the master thesis.

Kapitel 1

Einleitung

> Davis laughs. "The most complex print I've done had 120,000 layers in Illustrator. The printer called and said, 'How did you do this? How long did it take?' And I said, 'Oh, five minutes.' " [60]

Ohne Programmierung und die Rechenleistung seines Computers hätte Joshua Davis mit Sicherheit länger als fünf Minuten gebraucht, um diese 120.000 Ebenen in Illustrator zu erzeugen. Er schrieb einen Algorithmus, mit Hilfe dessen er der Maschine kommunizierte, was sie für ihn generieren soll. Diese Master Thesis beschäftigt sich mit ebendieser Kunst, bei der Algorithmen entworfen und auf Maschinen ausgeführt werden, um Kunstwerke zu schaffen.

Generative Kunst wird in den meisten Fällen mit Computern in Verbindung gebracht, weil diese auf den ersten Blick die Grundvoraussetzung dafür zu sein scheinen. Allerdings wurde generative Kunst bereits lange vor den ersten Computern produziert. Mit der Hilfe von Computern, die dafür geschaffen wurden, Prozesse abzuarbeiten und, wenn nötig, immer wieder exakt zu wiederholen, entwickelte sich allerdings auch die generative Kunst stärker als zuvor. Was zuvor mühsam manuell und mit viel Disziplin erschaffen werden musste, übernahmen dann computergesteuerte Maschinen. Die Definitionen über generative Kunst variieren vor allem in Bezug auf die verwendete Technologie. Ist der Computer eine Voraussetzung dafür oder nicht? Die verschiedenen Ansichten dazu werden in dieser Masterarbeit dargestellt und in Bezug zueinander gesetzt.

Außerdem divergieren die Meinungen, ob die interaktive Kunst ein Teilbereich der generativen Kunst ist oder ob sie eine eigenständige Form der Kunst darstellt. Die Definition interaktiver Kunst sowie ihre Beziehung zur generativen Kunst wird in der folgenden Arbeit ebenso behandelt.

Zielgruppe dieser Thesis sind sowohl Künstler und Designer, die sich für Technik und Programmierung interessieren, als auch Softwareentwickler, die Quellcode kreativ einsetzen möchten. Außerdem beschäftigt sich diese Arbeit

ebenso mit dem Quellcode als Medium und spricht somit auch Medientheoretiker an.

Die Motivation der Masterarbeit liegt in der Behandlung eines Themas, welches rationales Denken in Form von geschriebenem Quellcode und künstlerische Kreativität vereint. Es führt zu einer Verschmelzung zweier Bereiche, die sonst aufgrund ihrer unterschiedlichen Voraussetzungen stark voneinander abgegrenzt werden. Es bringt Gestalter hervor, die Quellcode implementieren und Programmierer, die künstlerisch tätig werden. Das ist es, was generative und interaktive Kunst in Zusammenhang mit der technologischen Entwicklung zunehmend interessant macht.

Inspiriert von dieser Master Thesis und den damit verbundenen Recherchen entstanden zahlreiche kleine Experimente, die sich generativer Elemente bedienen. Ein Auswahl dieser primär in Processing implementierten Experimente ist unter http://www.christian-schulze.eu/processing/ zu finden.

1.1 Zielsetzung

Die Intention dieser Master Thesis liegt in der Definition generativer sowie interaktiver Kunst und in der Darstellung ihrer historischen Entwicklung. Des Weiteren wird die Rolle des Quelltexts als Medium im klassischen Dreieck aus Künstler, Werk und Rezipient behandelt. Abschließend wird eine Werkanalyse des Projekts Ars Rata durchgeführt. Darin wird das Projekt in den Kontext der zuvor erläuterten Begrifflichkeiten gesetzt und die Frage behandelt, inwieweit Ars Rata ein generatives Kunstwerk ist.

1.2 Abgrenzung

Mit Ausnahme der historischen Entwicklung in Abschnitt 2.2 wird sich die Masterarbeit im Folgenden vorwiegend mit generativen Werken beschäftigen, die mit Hilfe von Computern erzeugt wurden. Wenn auch manche Definitionen generativer Kunst ebenso Werke zulassen, die ohne Computer erstellt wurden, liegt das Augenmerk dieser Thesis auf modernen generativen und interaktiven Werken, die nur noch selten ohne Computer funktionieren.

Ebenso konzentriert sich die Masterarbeit vorwiegend auf visuelle Werke, um das behandelte Thema in seiner Größe zu reduzieren und so eine vollständigere Abhandlung der übrigen Bereiche zu ermöglichen. Außerdem sind die aufgeführten Techniken größtenteils ebenso auf auditive Werke übertragbar und deshalb werden diese Werke in der Arbeit nicht explizit behandelt.

Der Bereich der Informationsvisualisierung kann ebenso als Teilbereich generativer und interaktiver Kunst gesehen werden. Allerdings würde eine vollständige Behandlung dieses Themas den Rahmen dieser Arbeit überschreiten. Deshalb wird an dieser Stelle darauf hingewiesen, dass Informationsvisualisierung in dieser Arbeit nicht behandelt wird.

Kapitel 2

Definition

Generative sowie interaktive Kunst sollen im folgenden Kapitel aus verschiedenen Perspektiven definiert und diskutiert werden. Des Weiteren wird die historische Entwicklung als zusätzliche Einführung in die generative Kunst präsentiert. Zum Abschluss des Definitionskapitels folgt eine Auseinandersetzung mit dem Zufall als Instrument generativer Kunst und dessen Bedeutung für die Gestaltung sowohl mit Computern als auch ohne sie.

2.1 Generative Kunst

Generative Kunst wird von Philip Galanter sehr stark vertreten. Er lehrt „generative art“ und „physical computing“ an der Texas A&M University und ist selbst ebenfalls als Künstler tätig (s. auch [43]). Seine Intention ist, generative Kunst populärer und stärker darauf aufmerksam zu machen, dass sie bereits mit der klassischen Kunst verwoben ist (vgl. [44, S. 8]). Er definiert sie in [45, S. 4] wie folgt:

> Generative art refers to any art practice where the artist uses a system, such as a set of natural language rules, a computer program, a machine, or other procedural invention, which is set into motion with some degree of autonomy contributing to or resulting in a completed work of art.

Wichtig ist ihm dabei, dass die Vorgehensweise des Künstlers bzw. das System entscheidend ist, in welchem er teilweise oder gänzlich die Kontrolle hat. Es geht in seiner Definition nicht darum, warum oder mit welchem Inhalt generative Kunst erzeugt wird, sondern nur um die Art und Weise, wie die Kunst gemacht wird. Das bedeutet ebenso, dass generative Kunst unabhängig von jeglicher Technologie, wie z. B. Computer oder andere Maschinen, ist.

Galanter präzisiert seine Formulierung später in [44, S. 1], um deutlich zu machen, dass ein Künstler zumindest einen Teil seiner Verantwortung an das von ihm geschaffene System abgeben muss:

> The key is that generative art happens when an artist chooses to cede some degree of control to an external system, and the artwork thus results from more than just the moment-to-moment intuitive decisions of the artist.

Auch in der Definition von Tjark Ihmels und Julia Riedel [49] ist von einem System die Rede, welches als Handlungsprinzip bezeichnet wird. Das Augenmerk liegt allerdings auf der Generierung neuer gestalterischer Inhalte:

> Die konsequente Anwendung eines vordefinierten Handlungsprinzips zum bewussten Ausschluss oder als Ersatz individueller ästhetischer Entscheidungen setzt die Generierung neuer gestalterischer Inhalte aus dafür bereitgestelltem Material in Gang.

Golan Levin, ein amerikanischer New Media Artist, der am MIT[1] auch Student unter John Maeda war, sieht in einem Interview von 2007 [55] generative Kunst als Teilbereich von New Media Art – der Kunst mit neuen Medien – und gibt einen Ausblick in die Zukunft:

> Generative art is one of many different subgenres of New Media art practice nowadays. [...] There are some people who work in the field of generative cinema: you hit a button and you get a different ending every time. That's a really active part of the field right now.

Das Feld des generativen bzw. interaktiven Kinos, das Levin anspricht, wird also in Zukunft an Bedeutung gewinnen. Hauptproblem dieses Feldes ist allerdings der enorme Produktionsaufwand der verschiedenen Varianten, die bei der Verwendung von Video- oder Filmmaterial für jede Variante separat aufgenommen werden müssen. Erst mit der Entwicklung von 3D-Engines, die in Echtzeit immer realistischere 3D-Renderings produzieren, kann dieses Problem behoben und zahlreiche Varianten über Programmcode generativ erzeugt werden.

In [17, S. xviii] setzt Matt Pearson generative Kunst zwischen Programmierung und Gestaltung und definiert sie als künstlerische Disziplin:

> Generative art is neither programming nor art, in their conventional sense. It's both and neither of these things. Programming is an interface between man and machine; it's a clean, logical discipline, with clearly defined aims. Art is an emotional subject, highly subjective and defying definition. Generative art is the meeting place between the two; it's the discipline of taking strict, cold, logical processes and subverting them into creating illogical, unpredictable, and expressive results.
>
> [...] Generative art is about creating the organic using the mechanical.

[1] Massachusetts Institute of Technology

Ein weiterer Künstler aus dem Bereich der generativen Kunst ist Marius Watz. Er beschreibt in [75] die Kernelemente aus der Sicht des Künstlers:

> A central aspect to the generative approach is the use of an externalized system, created by the artist but rarely completely under her control. [...] generative systems are dynamic processes that must be harnessed and even farmed. The artist specifies the initial boundaries and strategies of creation, and then enters into a feedback loop of adjusting parameters in a search for optimal regions in parameter space. The moment of genuine surprise is often the moment of breakthrough.

Im Widerspruch zu Galanter ist er in [74] der Meinung, dass generative Kunst untrennbar mit dem Computer verbunden, aber dennoch nicht ausschließlich auf Bildschirmausgaben beschränkt ist:

> While generative art is inextricably linked to the computer as a means of production, the work is not about the computer itself. While screen-based work and the investigation of realtime self-contained systems remain an important aspect of generative art, it would be a mistake to think generative work is primarily expressed in pixels. I for one look forward to an extended rethinking of computational aesthetics that encompasses a much wider range of possible outputs.

Watz macht außerdem auf das Problem aufmerksam, dass Theoretiker der Ansicht sind, generative Kunst sei das Gegenteil von Softwarekunst. Sie beschäftige sich nur mit Oberflächen und Ästhetik während sich Softwarekunst ebenso mit ihren Prozessen und politischen Aspekten beschäftigt. Wenngleich er dennoch gegen eine so strikte Trennung ist, muss Watz dieser Behauptung zustimmen. Es trifft zu, dass generative Kunst Software als gegeben hin nimmt und sich nicht mit ihrer militärischen, industriellen und kommerziellen Herkunft auseinander setzt. Watz ist der Meinung, dass generative Werke von Code geschaffen werden, aber nicht von Code handeln (vgl. [75]).

Im Gegensatz zu den bisherigen Definitionen sieht Rob Myers generative Kunst immer in Verbindung mit Technologie. Er behauptet, dass ein wesentlicher Teil des Werks durch einen nicht-menschlichen Prozess entstehen muss. Alles übrige bezeichnet Myers als prozedurale Kunst, die bis in die geometrischen Höhlenmalereien zurückgeht (vgl. [64]).

Außerdem betont Myers in [64] die Wichtigkeit der Eigenständigkeit der vom Künstler produzierten Software:

> The important fact is not the nature of the choice procedure, but that the choice is made autonomously by the software.

Myers kritisiert auch, dass generative Künstler den Quellcode ihrer Werke nicht preisgeben, sodass andere aufbauend neue kreative Werke daraus schaffen können. Er fordert eine wissenschaftliche Offenheit und freie Software, um den Fortschritt dieser Kunst zu gewährleisten (vgl. [64]).

Zusammenfassend kann man sagen, dass die Definitionen zunächst einmal belegen, dass generative Kunst keine Kunstrichtung, sondern eine Methode ist, Kunst zu produzieren. Es gibt keine spezifischen Merkmale in Bezug auf die Ästhetik und man kann generative Werke prinzipiell jeder Kunstrichtung zuschreiben, sofern das Werk mit generativer Methode kreiert wurde.

Außerdem liegt generativer Kunst immer ein System zugrunde, dass entweder manuell von einem Künstler befolgt oder automatisiert von einer Maschine bzw. einem Computer umgesetzt wird. Laut Myers ist ausschließlich die zweite Variante zulässig, weil nur sie ein tatsächlich autonomes System erlaubt, in das der Künstler keinen Eingriff mehr hat. Watz betont das ebenfalls mit seinem Hinweis auf ein externes System, worin der Künstler ausschließlich Grenzen definiert, aber nicht, was konkret innerhalb des Systems passiert. Diese Master Thesis basiert ebenfalls auf dieser modernen Definition, die ein externes System in Form von Technologie voraussetzt.

Ein generativer Künstler ist demnach eher ein Schöpfer eines Organismus, der im Idealfall ein Eigenleben entwickelt, mutiert und wächst. Der Vergleich mit künstlicher Intelligenz (Versuch, eine menschenähnliche Intelligenz nachzubilden, d. h., einen Computer zu bauen oder so zu programmieren, dass dieser eigenständig Probleme bearbeiten kann [54].) liegt an dieser Stelle sehr nahe und ist auch begründet. Mit Hilfe künstlicher Intelligenz wird versucht, menschliches Handeln zu simulieren. Ebenso spielt die Simulation von menschlichem Vorgehen in der generativen Kunst eine Rolle[2], jedoch ist es nur ein Teilbereich. In Kapitel 3 werden weitere Methoden generativer Kunst vorgestellt.

Ein weiterer wichtiger Punkt der modernen Definition im Vergleich zu klassischen Methoden, Kunst herzustellen, ist die Notwendigkeit von Algorithmen. Sie sind das Resultat des bereits angesprochenen Systems. Sie führen die Regeln des Systems aus. Allerdings müssen sie auch zuvor entwickelt und in eine für Maschinen verständliche Sprache, den Quellcode, übertragen werden. Weitere Ausführungen zum Quellcode folgen in Kapitel 4.

2.2 Historische Entwicklung

Im Sinne von Philip Galanters Worten – „generative art is as old as art itself“ [45, S. 12] – sollen Auszüge der geschichtlichen Entwicklung generativer Kunst folgen. Die Auswahl generativer Werke, ob mit oder ohne Maschinen, und die Interpretationsspielräume, wo generative Kunst beginnt und wo sie endet, sind groß. Dennoch zeigen diese Werke, dass generative Kunst als

[2] s. auch Cohens Aaron in Abschnitt 2.2.

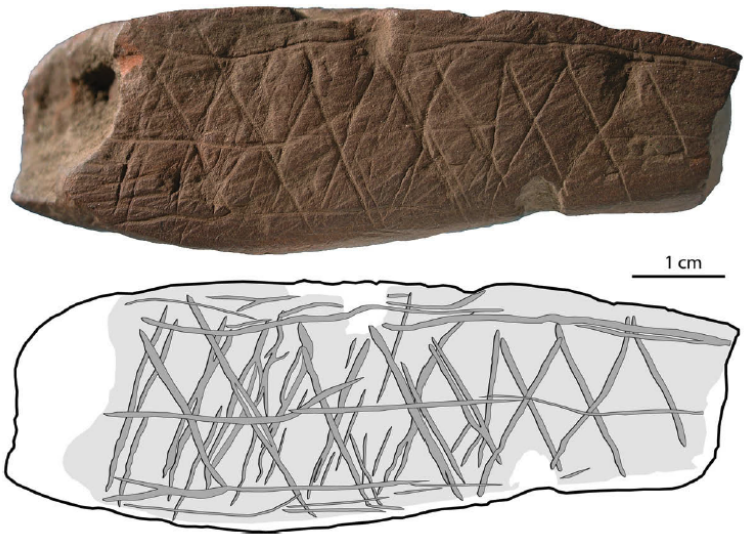


Abbildung 2.1: Gravuren aus der Mittelsteinzeit, Blombos Höhle, Südafrika, Foto und Spuraufzeichnung, Fundstück M1-6. Bildquelle: [11, S. 35].

Methode zwar nicht neu ist, aber auch, dass sie mit dem Computer endlich ein Medium gefunden hat, das ihrer Natur gerecht wird.

1999 und 2000 entdeckte ein Team um den Archäologen Christopher Henshilwood vom South African Museum in Cape Town handgroße Stücke aus rotem Ocker, die über 70.000 Jahre alt sind und mit eingeritzten Gittermustern aus Dreiecken verziert sind (Abb. 2.1). Es ist umstritten, ob diese Muster durch ihre Wiederholung bereits als generative Kunst bezeichnet werden können. Stanley Ambrose, Anthropologe an der University of Illinois, Urbana-Champaign sieht diese Verzierungen in [2, S. 247–249] aber bereits als Kunst an (vgl. [12] [45, S. 13]):

> This is clearly an intentionally incised abstract geometric design...It is art.

Auch Wolfgang Amadeus Mozart bediente sich bereits im 18. Jh. generativer Methoden indem er ein „musikalisches Würfelspiel“ entwarf. Es trägt den Untertitel: „Walzer oder Schleifer mit zwei Würfeln zu componieren ohne musikalisch zu seyn noch von der Composition etwas zu verstehen“. Mozart komponierte dafür 176 Takte Musik, die mit Hilfe der beiden Würfel kombiniert werden konnten und so 1116 verschiedene Kompositionen ergaben. Da diese Vorgehensweise auch nach Mozart noch öfter Anwendung fand, entwickelte die Kunstwissenschaft einen eigenen Begriff dafür: „aleatorische Musik“[3] (vgl. [40, 49] [45, S. 14]).

[3] Aleatorik (lat. alea, Würfel bzw. aleator, Würfelspieler)

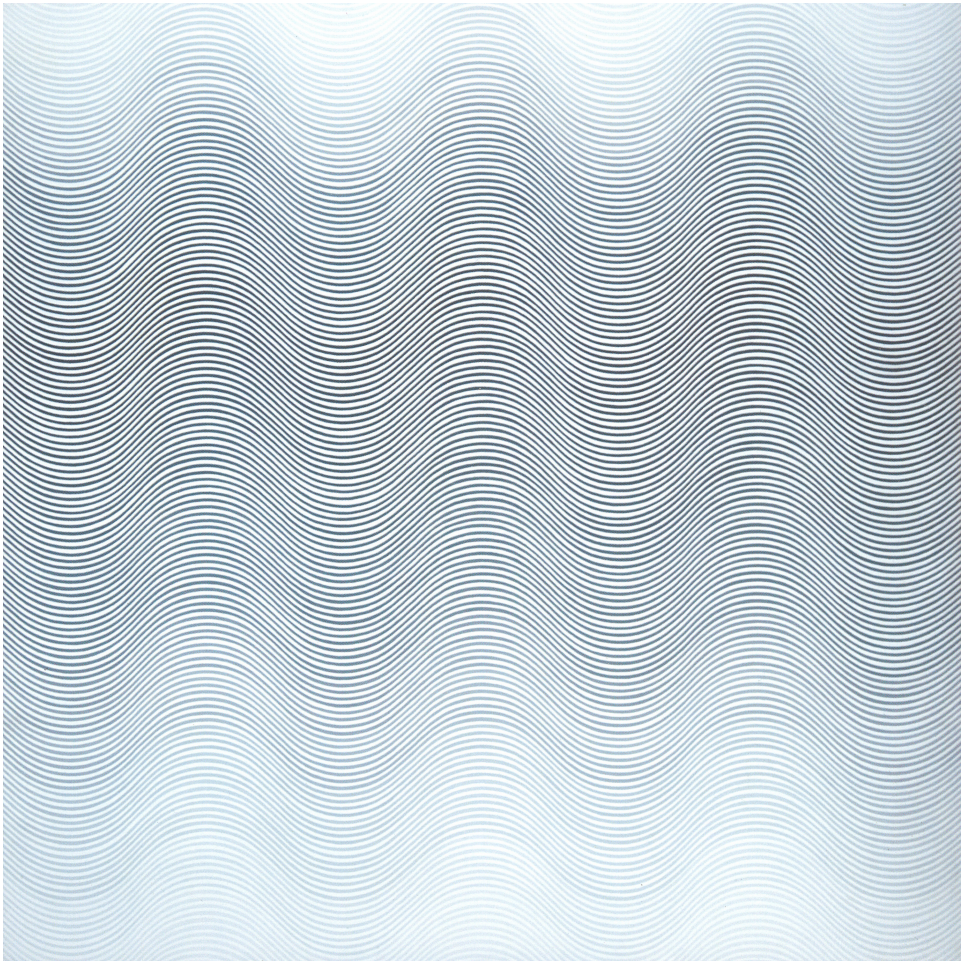


Abbildung 2.2: Bridget Riley: Polarity, Emulsion auf Leinwand, 177,8 × 177,8 cm, 1964. Bildquelle: [19, S. 48].

An dieser Stelle wird darauf hingewiesen, dass Rob Myers auch in dem musikalischen Würfelspiel keine generative Kunst sieht. Seine Begründung liegt nicht am Mangel von Zufall, welcher durch die Würfel gegeben ist, sondern weil der gesamte Prozess von einem Menschen ohne Zuhilfenahme eines Computers oder anderer Technologie durchgeführt wird (vgl. [64]).

Mit der optischen Kunst, auch Op-Art genannt, entwickelte sich in den 60er Jahren eine Stilrichtung, die durch Wiederholung abstrakte Muster und optische Effekte erzeugt. Bekannte Künstler der optischen Kunst sind beispielsweise Yaacov Agam, Richard Anuszkiewicz, Bridget Riley (Abb. 2.2), Jesús Rafael Soto und Victor Vasarely. Zur Erstellung ihrer Werke nutzten sie Algorithmen, obwohl sie noch keine Computer für die Produktion verwendeten. Die Abarbeitung der Algorithmen übernahmen die Künstler also

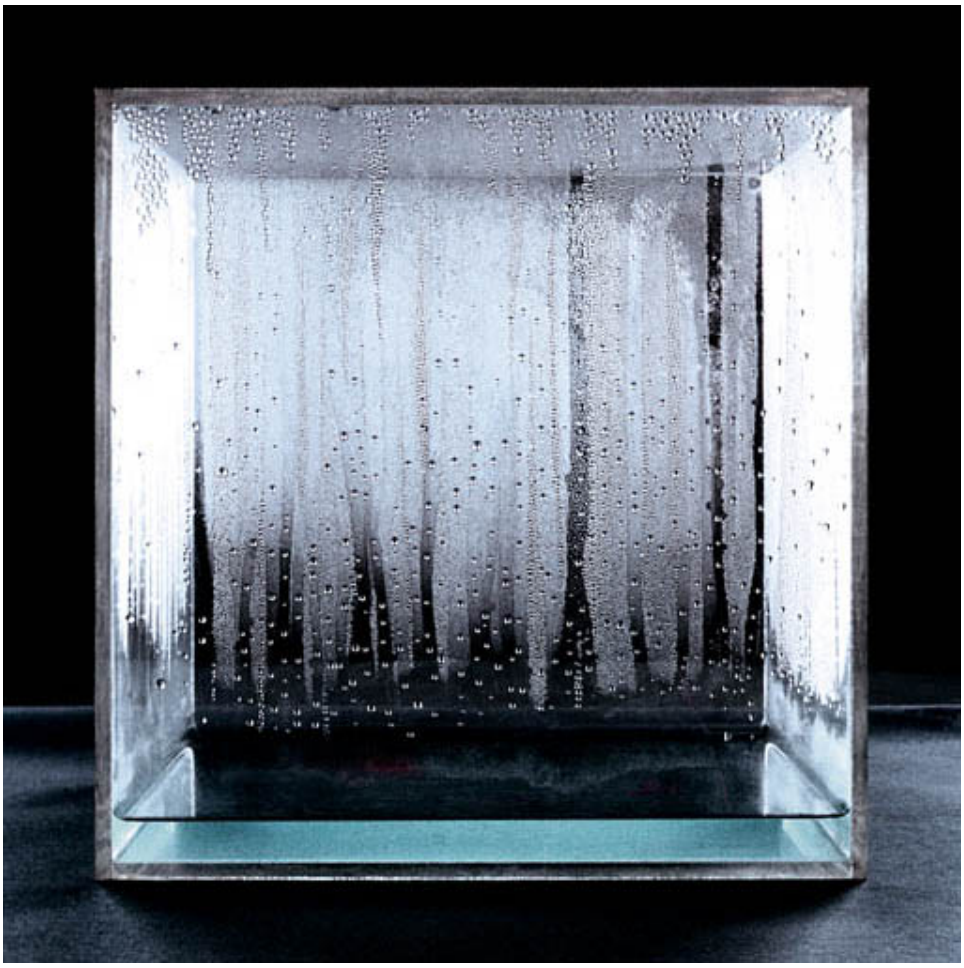


Abbildung 2.3: Hans Haacke: Condensation Cube, Acryl, Plastik und Wasser, 76 × 76 × 76 cm, 1963–65. Bildquelle: [37].

entweder selbst oder sie delegierten es an Assistenten (vgl. [19, S. 49] [65]).

Auch Hans Haackes „Condensation Cube“ (Abb. 2.3) von 1963–65 kann man als generatives Kunstwerk bezeichnen. Er besteht aus einem geschlossenen Würfel aus Plexiglas, der mit etwas Wasser gefüllt ist. Durch Temperaturschwankungen entsteht dann ein Wasserkreislauf aus Verdunstung und Kondensation. Dieses geschlossene System erinnert an eine einfache Wettersimulation, die u. a. durch ihre Unvorhersagbarkeit auch als generativ bezeichnet werden kann.

1965 definierte Max Bense den Begriff „generative Ästhetik“ in [3, S. 345] wie folgt:

> [...] die Zusammenfassung aller Operationen, Regeln und Theoreme [...], durch deren Anwendung auf eine Menge materialer

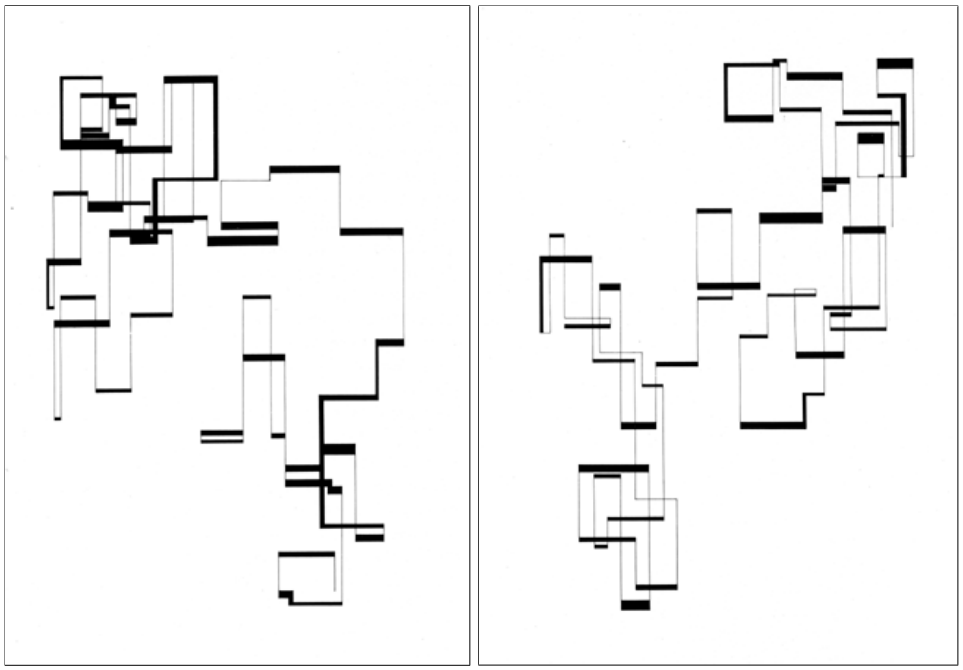


Abbildung 2.4: Manfred Mohr: P-10, „random walk“, Zeichenmaschine (Plotter) Tinte auf Papier, (2x) 50 × 35 cm, 1969. Bildquelle: [61].

> Elemente, die als Zeichen fungieren können, in diesen ästhetische Zustände (Verteilungen bzw. Gestaltungen) bewusst und methodisch erzeugbar sind.

Man erkennt bereits in dieser frühen Definition Elemente, die auch heute entscheidend für generative Kunst sind, zur damaligen Zeit jedoch noch nicht so einfach anzuwenden waren als es heute mit Hilfe von Computern der Fall ist. Bense versuchte damals durch detaillierte Analyse von realen Kunstwerken, einen rationalen Zugang zur Kunst zu finden und Systeme zu entdecken, nach denen man neue Bilder erschaffen kann (vgl. [49]).

Manfred Mohr begann ab 1969 seine Arbeiten nach systematischen Regeln zu produzieren und entwarf für jede Arbeit eigene Algorithmen in Fortran IV. Die Umsetzung erfolgte dann mit Hilfe eines Computers und eines Plotters (Abb. 2.4). Durch die Verwendung eigener Software ergaben sich laut [14, S. 95] folgende neue Möglichkeiten für die Künstler:

- Präzision als Teil ästhetischen Ausdrucks,
- hohe Geschwindigkeit in der Ausführung und dennoch Vielfältigkeit und Vergleichbarkeit unter den Werken,
- hunderte von Befehlen und statistischen Berechnungen sind möglich, die von einem Computer wesentlich leichter ausgeführt werden können als von einem Menschen, der nicht in der Lage ist, diese ganzen Befehle

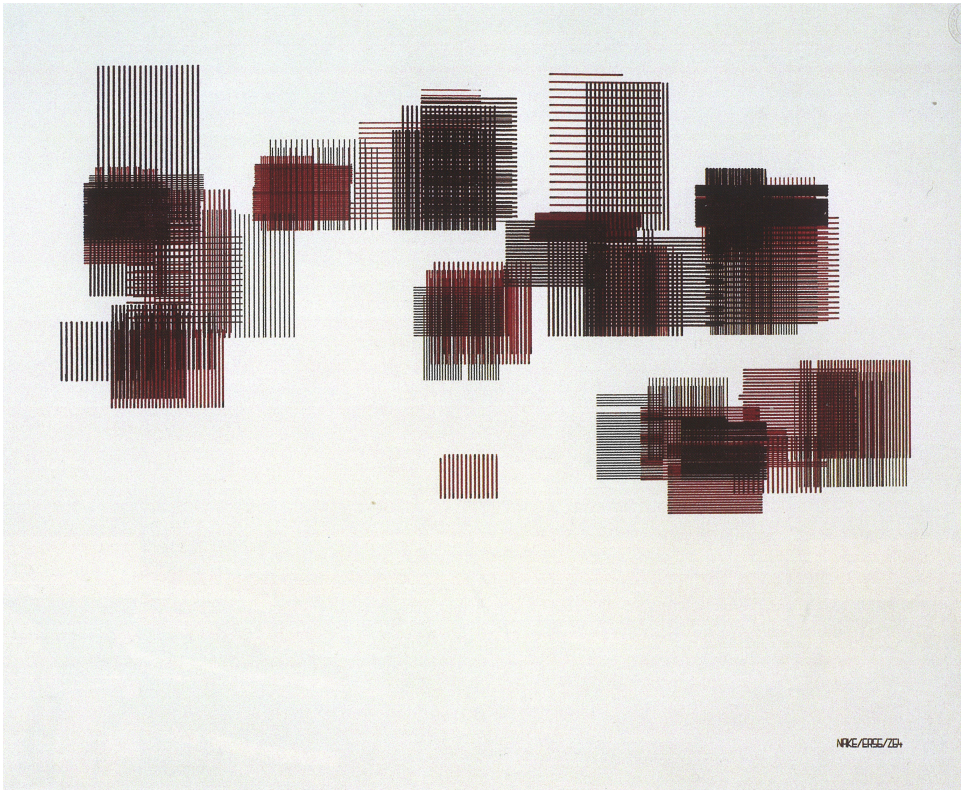


Abbildung 2.5: Frieder Nake: Felder von Rechteck Schraffuren überlagert, 1965. Bildquelle: [19, S. 52].

und Abfolgen im Gedächtnis zu behalten und

- der Dialog zwischen Mensch und Maschine gibt dem Künstler kontinuierliche Resonanz, die zu einem Lernprozess bei ihm führt, der ihm wiederum ein klareres Bild seiner Vorstellungen und Absichten vermittelt.

Mohr wird heute ebenso wie Jean-Pierre Hébert, Frieder Nake (Abb. 2.5) und Hans Dehlinger als Algorist bezeichnet (vgl. [61, 18]). Der Ursprung und die Erklärung dieses Begriffs ist denkbar einfach und wird von Roman Verostko, selbst Algorist, auf seiner Webseite [70] wie folgt definiert:

> [...] algorists are artists who create art using algorithmic procedures that include their own algorithms. [...] The jewel of algorist art lies in the artist's own "form-generating algorithms", the artist's unique procedures for creating the form.

In den 60er Jahren entwickelte sich unabhängig von der technischen Entwicklung eine neue Kunstrichtung – die Konzeptkunst. Sie wurde vor allem durch den amerikanischen Künstler Sol LeWitt geprägt (vgl. [53]), der diese

Kunst in [1, S. 14] wie folgt beschreibt:

> The idea becomes a machine that makes the art.

Wie der Name vermuten lässt, liegt das Augenmerk dieser Kunstrichtung nicht auf der Produktion eines fertigen Kunstwerks, sondern auf der Entwicklung eines Konzepts für ein Kunstwerk, eine Beschreibung oder Anleitung, wie es herzustellen ist. Dabei lässt das in Text geschriebene Konzept viel Interpretationsspielraum für den Maler oder Bildhauer, der es produziert. Das Konzept definiert ausschließlich einen Rahmen und eine Idee von einem Kunstwerk. Der Konzeptkünstler kann im Vorhinein allerdings nie exakt vorhersehen, wie das Ergebnis tatsächlich aussehen wird, weil es nicht in seiner Hand liegt. Diese Definition von Grenzen und das Überlassen der eigentlichen Umsetzung an ein externes System findet man auch in der generativen Kunst wieder, in der der Computer die Rolle des Malers oder Bildhauers übernimmt (Abb. 2.6).

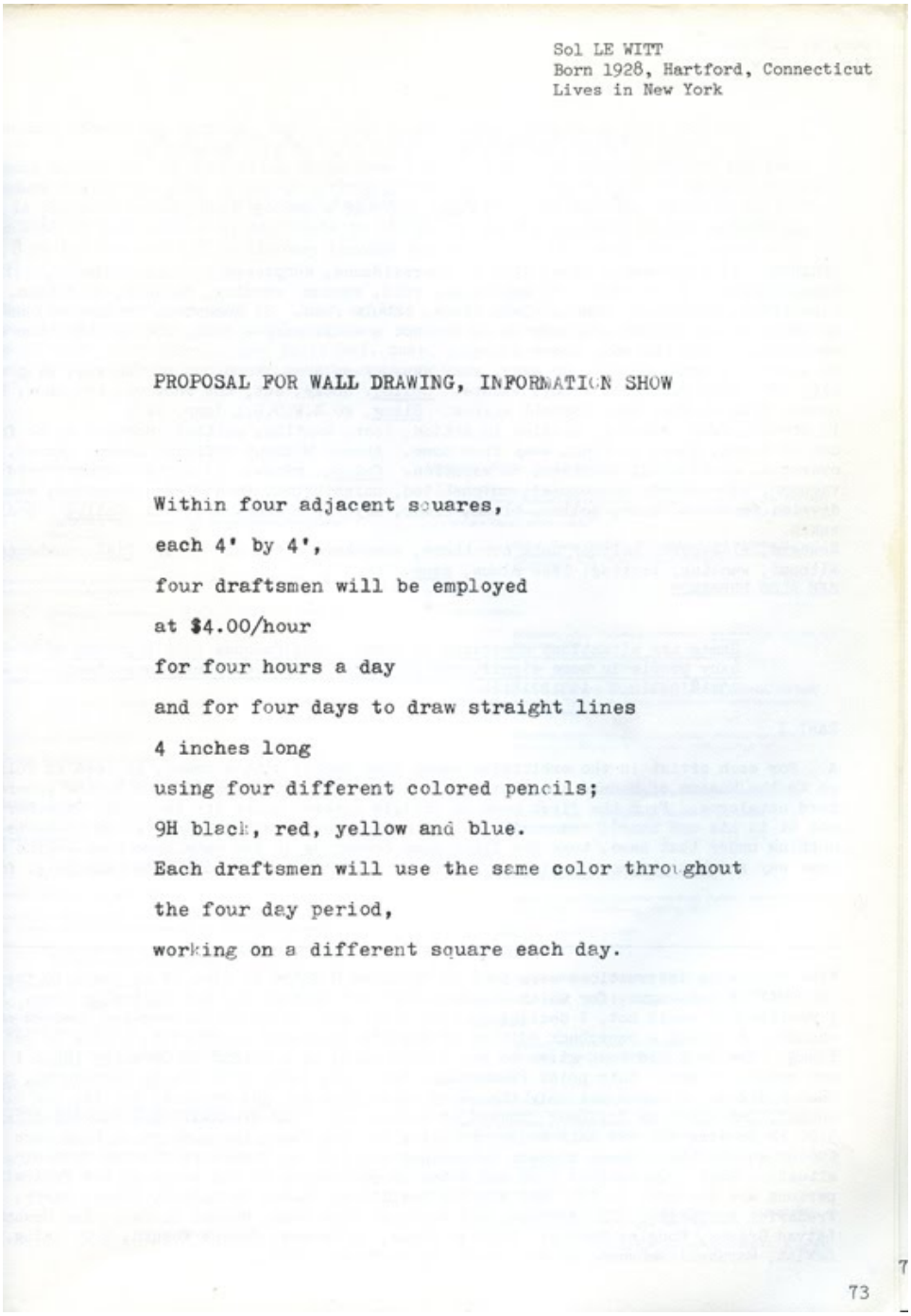
In einer Reihe von „simulation games“ implementierte 1970 der Mathematiker John Horton Conway das „Game of Life“. Es besteht aus einem zweidimensionalen Raster, welches im Idealfall unendliche Größe hat. Das Raster enthält zelluläre Automaten[4], die einen binären Status besitzen. Sie sind entweder lebendig oder tot. Dieser Status wird zu Beginn des Spiels festgesetzt. Danach folgt die Simulation des Spiels, die nach bestimmten Regeln abläuft. Diese Regeln beziehen sich auf den jeweiligen Status der Nachbarzellen jeder Zelle im Raster und bestimmen so ihren Folgestatus:

1. Haben lebende Zellen weniger als zwei lebende Nachbarn, sterben sie an Einsamkeit.
2. Haben lebende Zellen mehr als drei lebende Nachbarn, sterben sie an Überbevölkerung.
3. Eine lebende Zelle mit zwei oder drei lebenden Nachbarn überlebt in die Folgegeneration.
4. Eine tote Zelle mit genau drei Nachbarn wird in der Folgegeneration wiedergeboren.

Diese vier einfachen Regeln bestimmen das gesamte Spiel und können zu komplexen Strukturen führen, die für viele naturwissenschaftliche Bereiche, wie z. B. Biologie, Chemie, Physik und theoretische Informatik interessant sind (vgl. [46, 49] [22, S. 442]). In [49] wird deutlich, weshalb das „Game of Life“ zu seiner Zeit so bedeutend war:

> Die Faszination des Spiels lag sowohl in seiner Vielschichtigkeit als auch in seiner berechenbaren Unberechenbarkeit. „Game of Life“ bot eine Möglichkeit, grafische Aspekte einem fortdauernden System unterzuordnen und damit vom Individuum unabhängige

[4] s. auch Abschnitt 3.3.1.



Sol LE WITT
Born 1928, Hartford, Connecticut
Lives in New York

PROPOSAL FOR WALL DRAWING, INFORMATION SHOW

Within four adjacent squares,
each 4' by 4',
four draftsmen will be employed
at $4.00/hour
for four hours a day
and for four days to draw straight lines
4 inches long
using four different colored pencils;
9H black, red, yellow and blue.
Each draftsmen will use the same color throughout
the four day period,
working on a different square each day.

73

Abbildung 2.6: Sol LeWitt: Proposal for a Wall Drawing, Information Show, 1970. Bildquelle: [57].

Abbildung 2.7: Harold Cohen: Aaron's Garden, 76,2 × 55,88 cm, 1989. Bildquelle: [70].

> Formen zu generieren, indem ein dem Lebenszyklus nachempfundener Algorithmus zugrunde gelegt wurde.

1973 entwickelte Harold Cohen ein Computerprogramm namens „Aaron", welches er bis heute noch weiterentwickelt (Abb. 2.7). Es besteht aus ca. 1,5 Megabyte LISP Code („eine Familie von Programmiersprachen, die 1958 erstmals spezifiziert wurde" [59]) und einer Apparatur, die mit verschiedenen Pinseln und später auch Farbe malte. Cohen brachte Aaron ein komplexes Regelwerk bei, das „allgemeingültige ästhetische Anforderungen an Linien, Flächen, Formen und Farben formulierte und die Aufteilung eines Blattes analysierte." [49] Neben diesen formalen Anforderungen integrierte Cohen über die Jahre auch immer mehr Motive von Menschen, Pflanzen, Bäumen und einfache Objekte wie Kisten und Tische. Aaron kann also nur Dinge malen, die er vorher gelernt hat, allerdings variiert er dann innerhalb dieser Motive in Form und Farbe (vgl. [23, 25, 49, 19]). Es gibt zahlreiche Diskussionen darüber, inwieweit Aaron eigenständige und tatsächlich kreative Bilder produziert, Cohen hat darauf in [25] allerdings eine passende Antwort:

> [...] the museums thought it was art. People buy it as if it's art. They use it as if it's art – they put it in frames and put it on the wall.

2.3 Interaktive Kunst

Definiert ist Interaktion als eine Wechselbeziehung zwischen Handlungspartnern. Dabei können diese Handlungspartner Individuen, Gruppen oder Institutionen sein (vgl. [21, S. 110]). Bereits zur Zeit des Dadaismus interagierten die Künstler in sogenannten „Happenings“[5] mit dem Publikum (vgl. [48]).

Georg Trogemann trennt zunächst die Begriffe Interaktivität und Interaktion in seinen Ausführungen. Er sieht in Interaktivität nur „die technische Fähigkeit eines Programms, während seiner Laufzeit neue Eingaben zu verarbeiten“ [22, S. 108], wohingegen er Interaktion vorerst als ursprünglichen Begriff der Soziologie definiert, der „das aufeinander bezogene Handeln von zwei oder mehreren Personen beschreibt.“ [22, S. 108] Allerdings bezieht sich Interaktion laut Trogemann ebenso auf die Kommunikation zwischen Mensch und Maschine, was in [22, S. 108f] deutlich wird:

> Interaktive Anwendungen zielen immer schon auf symbiotischen Nutzen durch Interaktion ab, d. h. das hybride System aus Mensch und Maschine soll als Gesamtsystem in der Lage sein, Aufgaben zu bewältigen, die der Mensch alleine nicht in Angriff nehmen könnte.

Auch deshalb bezeichnet Trogemann den Computer als „hybride Maschine“ [22, S. 88]. Er sieht die heutigen Arbeitsprozesse am Computer als symbiotisch an, in denen „Mensch und Maschine im interaktiven Zusammenspiel“ [22, S. 88] Probleme lösen, die sie als Teilsystem allein nicht lösen könnten.

Trogemann verweist auch auf Peter Wegner, der in [76] einen Paradigmenwechsel von Algorithmen zu Interaktion beschreibt. Wegner will damit auf den Bedeutungswandel des Algorithmusbegriffs aufmerksam machen. Die wesentlichsten Eigenschaften in der Definition eines Algorithmus sind Determiniertheit und Determinismus, d. h. er muss bei gleicher Eingabe immer das gleiche Ergebnis liefern (Determiniertheit) und zu jedem Zeitpunkt dürfen nur definierte und reproduzierbare Zustände auftreten (Determinismus). Es steht also von Beginn an fest, welche Befehle aufeinander folgen und diese Abfolge wird immer die gleiche sein (vgl. [24]). Diese Definition ermöglicht laut Wegner allerdings keine externen Eingaben, die diese Abfolge verändern könnten. Deshalb arbeiten seine „Interaction Machines“ nicht linear die gegebenen Befehle ab, sondern reagieren interaktiv auf Veränderungen durch beispielsweise Benutzer der Maschine (vgl. [76]). Trogemann bewertet dieses neue Modell in [22, S. 109] wie folgt:

> Wichtig an Wegners neuem Maschinenmodell ist, dass es – auch wenn es den Rahmen der klassischen Algorithmentheorie nicht sprengt – die derzeitige Praxis der Maschine sehr viel besser abbildet als die klassischen Modelle, die den Begriff der Interaktivi-

[5] engl., Ereignis, Geschehnis.

> tät nicht kennen. Interaktionsmaschinen bieten eine praktikable Möglichkeit, die Interaktion in objektorientierten und verteilten Systemen angemessen zu beschreiben.

Generative Werke können ebenso interaktiv sein, sich also dynamisch in Abhängigkeit von verschiedenen Parametern verändern. Es lässt sich nun allerdings darüber streiten, ob interaktive Kunst bzw. generative Kunst mit interaktivem Anteil ein Teilbereich der generativen Kunst ist oder eigenständig behandelt werden sollte. Golan Levin sieht interaktive Kunst als Teilbereich und beschreibt es in [56] folgendermaßen:

> Some generative art operates completely autonomously, while some generative artworks also incorporate inputs from a user, or from the environment.

Im Gegensatz dazu trennt Gruber in [10, S. 24] die interaktive Kunst von der generativen und definiert sie wie folgt:

> Interaktive Kunst bezieht über den Produktionsprozess hinaus den Rezipienten als aktiv gestalterisches Element in das Werk ein und eröffnet sich dem Publikum erst nach einer gewissen Explorationsphase. Diese Werke nutzen Peripheriegeräte um auf verschiedene Parameter wie Bewegung, Druck, Geräusche oder Licht zu reagieren und sich entsprechend ihrer Programmierung zu verändern. Interaktive Kunst etabliert, im Gegensatz zur Generativen Kunst, immer einen Dialog zwischen Kunstwerk und Besucher. Das Publikum ist eingeladen mit der Arbeit zu kommunizieren und diese spielerisch zu erforschen.

Auch Marius Watz sieht in [74] die interaktive Kunst getrennt von der generativen Kunst, die er als geschlossenes System betrachtet:

> Interactive art exploits the feedback loop of interaction between a system and its user(s), with custom software systems generally considered a necessary evil rather than an end in itself. Generative art is primarily interested in closed system, self-contained constructs investigated for their formal and material qualities. This might seem like a trivial difference, but it places the concerns of generative art closer to traditions of drawing or painting than to the relational aesthetics so common in the media art field.

Interaktion kann auf sehr vielen Ebenen stattfinden, die nicht zwangsläufig generativ sein müssen. Deshalb ist ein Kompromiss aus beiden Definitionen die bessere Lösung. Sie sind beide eigenständige Möglichkeiten, Kunst zu produzieren, aber es gibt eine Schnittmenge (Abb. 2.8), in der generative Kunst interaktiv und interaktive Kunst generativ ist. Generative sowie interaktive

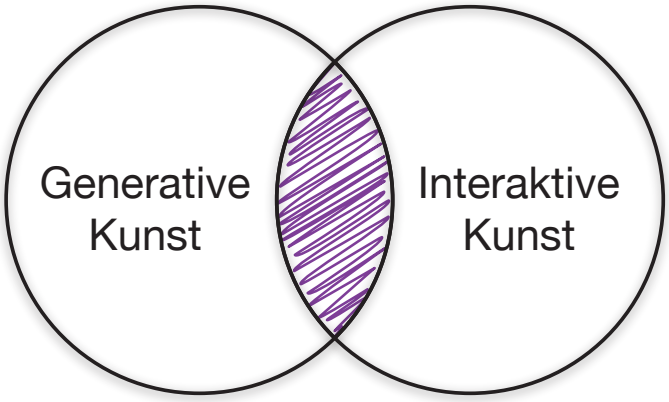


Abbildung 2.8: Schnittmenge von generativer und interaktiver Kunst.

Werke innerhalb der Schnittmenge können somit auch als Weiterentwicklung der beiden Teilmengen gesehen werden. Durch interaktive Anteile kann generative Kunst ebenso einen Mehrwert erhalten wie interaktive Kunst, die um generative Elemente erweitert wird. Dennoch sind diese Ausprägungen unterschiedlich. Während die Erweiterung um interaktive Elemente z. B. durch die Parametrisierung[6] in generativer Kunst meist einen geringen Aufwand erfordert, kann dieser bei interaktiver Kunst wesentlich größer sein, da generative Elemente erst erfunden werden müssen. Außerdem könnte die Integration eine erhebliche Abweichung vom ursprünglichen Werk bedeuten. Das Hinzufügen interaktiver Elemente wie z. B. Schieberegler oder Schalter muss im Gegensatz das eigentliche Werk nicht beeinflussen, sondern kann es beispielsweise innerhalb gesetzter Grenzen lediglich variieren.

Dieter Daniels definiert in [9] und [13, S. 91ff] sieben Varianten interaktiver Werke, die im Anschluss näher erläutert werden:

1. Interaktion mit einer Videostory, deren Ablauf vom Betrachter bestimmt wird.
2. Das geschlossene System einer Datenwelt, die vom Betrachter erkundet wird.
3. Interaktion zwischen Körper und Datenwelt.
4. Datensystem mit Eigendynamik, die sich durch die Interaktion weiterentwickelt.
5. Dialogische Modelle.

[6] s. auch Abschnitt 3.4.

6. Der „exemplarische Betrachter“.
7. Ansätze zu einer Kollektivität im Medienraum.

In Punkt eins entsteht die Interaktion aus der Möglichkeit, einen Film als Betrachter aktiv zu beeinflussen. Er kann an vordefinierten Stellen selbst entscheiden, wie sich die Story entwickeln soll. Bereits in den 80er Jahren gab es erste Werke von Grahame Weinbrens („The Erl King“ 1986, „Sonata“ 1991/93) und Lynn Hershman („Deep Contact“ 1989/90, „A Room of One's Own“ 1992) zu diesem Thema (vgl. [9] [13, S. 91f]).

Probleme solcher Interaktionen liegen dabei sowohl in der Produktion als auch in der Distribution. Die Story sollte möglichst dynamisch sein und viele Eventualitäten abdecken, sodass die Auflösung stets im Ungewissen bleibt. Je mehr Dynamik die Story bietet, desto immersiver wird das Erlebnis des Betrachters. Allerdings bringt das einen sehr hohen Produktionsaufwand mit sich, da alle möglichen Entscheidungen, auch die, für die sich der Betrachter nicht entscheidet, aufgenommen werden müssen. Des Weiteren liegt die technische Herausforderung in der Veröffentlichung des Werks. Die großen Datenmengen der Videos konnten in den 80er Jahren kaum auf ein geeignetes Medium kopiert und abgespielt werden. Erst mit der Entwicklung der DVD und nunmehr auch der BluRay erweiterten sich die Möglichkeiten. Auch virtueller Speicher auf Servern bietet sich an. Beispielsweise können so interaktive Videos auf Internet-Videoplattformen publiziert und in einzelnen Clips verbunden werden.

Punkt zwei spricht laut Daniels vom klassischen Grundmodell der 3D-Interaktion, wie sie z. B. Jeffrey Shaws in seinen Installationen („The Legible City“ 1988, „The Virtual Museum“ 1991) eingesetzt hat. Der Rezipient hat hierbei die Möglichkeit, eine Datenlandschaft zu erkunden, die allerdings unveränderbar ist (vgl. [9] [13, S. 92f]). Das Ziel dieser Installationen ist, eine möglichst hohe Immersion beim Betrachter zu schaffen. Gleichzeitig ist das auch die höchste Herausforderung, denn die Immersion hängt am meisten vom vermittelten Realismus der Datenlandschaft ab. Umso realistischer also die Immersion ist, desto mehr glaubt der Rezipient, sich tatsächlich in dieser Datenlandschaft zu befinden.

Daniels spricht in Punkt drei alle Techniken der virtuellen Realität an, die eine Erweiterung der Wahrnehmung bedeuten indem sie eine Verbindung von Datenstruktur und Körper herstellen. Als Beispiel führt er das Werk „Zerseher“ (Abb. 2.9) als ein Projekt der Berliner Firma ART+COM auf (vgl. [9]). Es geht dabei um ein Gemälde, welches sich verfremdet, sobald eine Person es betrachtet. Dabei werden genau die Stellen verzerrt, auf die der Betrachter schaut. Realisiert wurde es mit einem Monitor und einem Eye-Tracker, die in der Wand platziert und über einen Computer verbunden wurden. Die Interaktion beruht auf der Sammlung von Daten, die der Rezipient aussendet, in diesem Fall die Augenbewegung.

Im nächsten Punkt wird ein Datensystem mit Eigendynamik themati-

Abbildung 2.9: ART+COM: Zerseher, 1992. Bildquelle: [30].

siert, die durch den Rezipienten variiert werden kann (vgl. [9] [13, S. 93f]). Es liegen also bereits Daten vor, die eine gewisse Art von Lebendigkeit simulieren. Greift der Rezipient in das Leben ein, passt sich das System an die neuen Voraussetzungen an. Es ist also ebenfalls möglich, dass das System eine Art künstliche Intelligenz ist, die in der Lage ist, dazu zu lernen. Diese These ist eng verbunden mit den Forschungen von Alan Turing, der sich aus der wissenschaftlichen Sicht mit künstlicher Intelligenz befasste. Er schlug 1950 den Turing-Test vor, um die künstliche Intelligenz einer Maschine mit dem Denkvermögen des Menschen zu vergleichen. Der Test gilt als bestanden, wenn ein Mensch nach einer intensiven Befragung per Texteingabe immer noch nicht genau sagen kann, ob sein Gegenüber ein Mensch ist oder nicht. Der totale Turing-Test verwendet zusätzlich noch ein Videosignal (vgl. [69], s. auch [22, S. 244ff]).

Punkt fünf beschreibt Konzepte, die laut Daniels die Interaktion zwischen Mensch und Maschine um ein Medium erweitern, das zwischen ihnen steht. Das Medium kann, wie es Douglas Davis bereits in den 70er Jahren nutzte, eine Fernsehübertragung („Talk Out: A Telethon“ 1972, s. auch [41]), also eine Live-Übertragung zwischen zwei Ausstellungsorten oder aber auch Videobänder („The Austrian Tapes“ 1974) sein, die interaktiv ausgewählt und abgespielt werden. Ebenso können Interfaces ein Medium darstellen, das zwischen den Interagierenden steht und virtuelle Gesten mit realer Mimik kombiniert (vgl. [9]).

Im folgenden Punkt ist vom „exemplarischen Betrachter“ die Rede. Mit ihm definiert Daniels den eigentlichen Akteur innerhalb einer interaktiven Installation. Er ist einer von vielen Rezipienten, der tatsächlich mit dem Kunstwerk interagiert und nicht, wie bei jeder anderen Kunst, nur konsumiert. Gleichzeitig kritisiert Daniels die Einschränkung vieler interaktiver Installationen auf ebendiesen einen exemplarischen Betrachter. Er behauptet, es erzeuge Einsamkeit beim Rezipienten, weil er nicht mehr nur ein Betrachter unter vielen ist, die sich als Gruppe um das Werk versammeln (vgl. [9] [13, S. 95f]). Er wird stattdessen zum Individuum, das von den restlichen Betrachtern ebenso wie das Werk beobachtet wird. Vielen Rezipienten ist der Verlust dieser Anonymität der Gruppe unangenehm genug, um nicht mit dem Werk zu interagieren. Es müssen also Möglichkeiten gefunden werden,

die es den Rezipienten ermöglichen, Teil des Kunstwerks zu werden, ohne spezifische Aufmerksamkeit der anderen Rezipienten zu erhalten, sofern sie es nicht wünschen. So würde jeder Rezipient die Kontrolle über seine Anonymität behalten. Er könnte mit dem Werk interagieren und hätte dennoch die Möglichkeit, sein interaktives Erlebnis mit anderen Besuchern zu teilen.

Daniels' letzter Punkt mit Ansätzen zu einer Kollektivität im Medienraum spricht exakt diesen Punkt an, der Einsamkeit des exemplarischen Betrachters entgegen zu wirken. Möglich wird das durch komplexere Kommunikationsstrukturen wie unser heutiges Internet (vgl. [9] [13, S. 94f]). Es ermöglicht vernetztes Schreiben und soziale Interaktion in Texten, Bildern sowie Videos und Ton. Es vereint sämtliche Medien und erweitert diese zusätzlich mit Applikationen, die ebenfalls interaktiv sowohl im funktionalen als auch im künstlerischen Kontext auf die Inhalte reagieren.

Das Internet hat sich zu einem weiteren wichtigen Medium entwickelt, in dem die Interaktion eine wesentliche Rolle spielt. Auch Dieter Daniels beschreibt in [9] und [13, S. 97] diesen Prozess vom Internet als Kommunikationsmedium zum Konsum- und Broadcastmedium:

> Die einschneidende Veränderung der Denkmodelle zeigt sich an dem Bedeutungswandel der zentralen Begriffe. „Cyberspace" wird nicht mehr vorrangig als Projektion des realen Raums und des menschlichen Körpers in den Datenraum verstanden, sondern als Vernetzung aller Kommunikationsstrukturen. „Interaktivität" wird von der Mensch-Maschine-Interaktion wieder zur zwischenmenschlichen Interaktion, deren Strukturen durch die Übermaschine des Internets mit seinen Millionen von angeschlossen Computern bzw. Nutzern geprägt wird.

Die Menschen selbst stehen also im Zentrum und interagieren mit Hilfe der Technik. Doch auch die Technik selbst beeinflusst die Kommunikation. Ein gutes Beispiel dafür ist Facebook, dessen medientheoretische Definition an sich schon interessant ist. Es existiert, wie jede Website, im Gegensatz zu den klassischen Medien allein digital, im Rechner, auf Servern, die das Internet bestimmen. Es ist multimedial durch Texte, Fotos und Videos, aber vor allem ist es sozial - eine Komponente, die kein anderes Medium so stark einfließen lässt. „User-driven-Content" bezeichnet man diese Statusmeldungen, Nachrichten und Fotos der Benutzer, die Facebook erst zum Leben erwecken. Facebook selbst stellt nur die Plattform dafür zur Verfügung und ist somit Kommunikationsmedium zwischen den Benutzern. Dieses Kommunikationsmedium besteht ausschließlich aus Programmcode, aus Algorithmen die von Maschinen, den Facebook-Servern, ausgeführt werden. Und diese Algorithmen bestimmen, was den menschlichen Benutzern an Informationen bereitgestellt oder vorenthalten wird. Dabei ist nicht nur entscheidend, wer diese Information auf der Plattform bereitstellt, ob es z. B. ein Freund ist oder eine fremde Person, es spielt z. B. ebenso eine Rolle, wie Facebook die

Beziehung zwischen zwei Benutzern beurteilt. Demnach werden beispielsweise Statusmeldungen von Freunden, mit denen oft kommuniziert wird, bevorzugt präsentiert. Hingegen werden Meldungen vermeintlicher Pseudo-Freunde, mit denen kaum oder gar keine Kommunikation in Form von Nachrichten, Kommentaren, etc. betrieben wird, herausgefiltert. So entscheidet also das Medium selbst, welche Informationen tatsächlich bei den Nutzern ankommen und welche nicht. Es denkt und interpretiert. Es urteilt über den Status der Beziehung zwischen zwei Benutzern.

Die mit dem Internet entstandene Netzkunst erregte bisher kaum Aufmerksamkeit. Daniels begründet das mit „dem wechselseitigen Ausschlußprozeß zwischen Kunstdiskurs und Netzkulturdiskurs“ [9]. Sie werfen sich gegenseitig „Kommerzabhängigkeit, Pseudo-Progressivität, Schein-Offenheit oder Borniertheit“ [9] vor und machen die Netzkunst so zur „Randgruppe zweier Randgruppen“ [9]. Außerdem löst das Internet, wie auch andere Massenmedien, alle Kontextbezüge auf, die Kunst der Moderne aber ist im 20. Jahrhundert immer kontext-spezifischer und -abhängiger geworden. Laut Daniels steht die Kunst im Internet „deshalb vor dem Dilemma sich medial an alle, aber kontextuell an niemand zu wenden.“ [9]

Außerdem verweist er auf den Prix Ars Electronica des Jahres 1999 in der Kategorie „.net“, der an das quelloffene Betriebssystem Linux vergeben wurde. Damit zeigt die Jury eines Kunstpreises, „daß Programmierung die eigentliche Kunst sei, und was Künstler damit machen zweitrangig“ [9]. Doch tatsächlich geht es darum, dass Kunst und Technik nicht weiterhin als getrennte Bereiche betrachtet werden sollen, sondern stattdessen eine „Ganzheit von Kulturleistung, Technikinnovation und Naturforschung“ [9] angestrebt wird. Technik ist also als ein unverzichtbarer Teil der modernen Kunst anzusehen, was vor allem auch für die interaktive Kunst von Bedeutung ist (vgl. [9]).

Ingo Wörth, Cornelia Hochmayr und Katja Kwastek haben sich im Kontext der Ars Electronica 2007 mit der Rezeption interaktiver Kunst beschäftigt und Künstler-Interviews, Gruppendiskussionen, Beobachtungen sowie Befragungen durchgeführt (vgl. [63, S. 111]). Die Ergebnisse ihrer Forschungen werden in den folgenden Abschnitten aufgeführt.

Bedienungsanleitungen oder Schilder, die eine Installation erläutern, sind von den Künstlern nicht erwünscht. Sie sehen es als eine großen Teil des Prozesses interaktiver Kunst an, dass der Rezipient das Kunstwerk entdeckt und von selbst erschließt. Außerdem stellen Anleitungen eine Vorschrift der Verwendung dar und verhindern so die eigenständige Interpretation durch den Rezipienten. Stattdessen werden Vermittler bevorzugt, die vor Ort Hilfestellung geben und Rezipienten zur Interaktion motivieren indem sie beispielsweise Interaktionsmöglichkeiten aufzeigen. Dadurch können Barrieren überwunden werden, die z. B. durch die Technologie, Sprache der Installation oder das klassische Verbot, nichts anfassen zu dürfen, entstehen (vgl. [63, S. 111]).

Die Künstler erwarten zum Großteil keine fixe Interpretation ihrer In-

stallationen. Sie sind der Meinung, dass sich die Bedeutung des Kunstwerks für jeden Rezipienten individuell erschließt, wenn auch der Künstler bei der Konzeption eine konkrete Idee oder ein übergeordnetes Thema verfolgte. Dennoch ist die Interaktion selbst der wichtigere Aspekt, wie es auch in [63, S. 111] zur Geltung kommt (vgl. [63, S. 111]):

> Allerdings überwiegt auch hier der interpretative, prozesshafte Charakter interaktiver Kunst, da das Werk seinen letzten Teil erst in der Interaktion mit RezipientInnen findet. Diese reagieren unvorhersehbar und unterschiedlich auf die Installationen – und das ist von den KünstlerInnen durchaus so gedacht.

Außerdem ergab sich aus den durch die Forschungsarbeit initiierten Diskussionen, „dass sich in der interaktiven Kunst ein Wandel von der Kunst als Produkt zur Kunst als Prozess vollzogen hat, der dem Publikum ein breites Spektrum an Partizipationsmöglichkeiten bis hin zur Mitautorenschaft am Kunstwerk bietet“ [63, S. 112]. Des Weiteren ging aus den Gruppendiskussionen der Forschungsarbeit hervor, dass auch der Präsentationsraum einen Einfluss auf die Installation hat und umgekehrt auch die gezeigten Objekte die Wahrnehmung des Raumes beeinflussen (vgl. [63, S. 113]).

Abschließend wird im Resumee der Studie nochmals die hohe Relevanz interaktiver Kunst als Ausdrucksmittel in der Verbindung von Gesellschaft, Kunst und Technik hingewiesen. Außerdem definieren die Forscher interaktive Kunst als einen vom Konzept her sozialen Prozess, der vor allem durch Vermittlung bzw. Heranführen noch stärker gefördert werden kann (vgl. [63, S. 147]).

2.4 Zufall

Eine gute Einleitung in die Thematik des Zufalls bietet Dietmar Guderian in [15, S. 10]:

> In fast jeder Epoche werden die beinahe gleichen existentiellen Grundfragen um das Problem Zufall neu gestellt. Zum Grundkanon des philosophisch weltanschaulichen Aspekts von Zufall gehören die Themen-Komplexe: Kausalitätsprinzip der Weltordnung und Rolle des Zufalls, Sinndeutung des Zufalls, Verhalten des Menschen gegenüber einem sein kausales Handeln und Denken durchkreuzenden Zufall, Freiheit und Notwendigkeit, Schicksal, Tragik und Katastrophe.

Mit der Zeit hat sich also auch der Begriff und die Bedeutung des Zufalls gewandelt. In der Antike nannte man den Zufall „Tyce“, „ein Zusammentreffen der besonderen Art“. Aristoteles spricht beispielsweise von einem gelenkten Zufall in Bezug auf die Statue des Mitys, mit der er versucht, den Schein

der Absichtlichkeit im Zufall zu beweisen. Die „Tyce“ wird später immer mehr als grundloser Zufall aufgefasst und wird somit auch als unglaubwürdiger betrachtet. Die Christen kritisieren vor allem das „Wertindifferente und Absichtslose“ des antiken Zufalls. Willkür passt nicht in ein christliches Weltbild, das einen unfehlbaren und lenkenden Gott im Beginn eines jeden Ereignisses sieht. Im späteren Mittelalter wandelt sich die Auffassung des Zufalls wie in der Spätantike zum Wunder. Schließlich kommt es in der Neuzeit zu vielen verschiedenen Zufallsdeutungen, weil die Kirche in deren Weltbild zunehmend an Macht verliert (vgl. [15, S. 10]).

In der Kunst des 19. und 20. Jahrhunderts wird die Foto-, Film- und Videotechnik genutzt, um realistische Lebensauschnitte zufällig aneinander zu fügen. Dadaist Hans Arp entdeckte für sich um 1916 und 1917 das Prinzip des Zufalls durch einen Zufall als er aus Unzufriedenheit eine Zeichnung zerriss und anschließend begann, die am Boden liegenden Stücke in einer Collage zu arrangieren. In mehreren Collagen (Abb. 2.10), die er mit „nach den Gesetzen des Zufalls geordnet“ betitelte, schnitt er dann Farbpapiere in annähernde Quadrate und lies diese anschließend auf einen Karton fallen. So ergab sich eine zufällige Anordnung der Farbpapiere, die Arp jedoch anschließend noch korrigierte. Er nutzte die zufällige Konstellation somit nur als Inspiration. Dennoch war er einer der ersten Künstler, die mit Hilfe des Zufalls „den Mythos vom genialen Künstlerindividuum zu unterlaufen“ [6, S. 30] versuchten (vgl. [6, S. 30] [15, S. 12]). Guderian schreibt dazu in [15, S. 12]:

> Zufall liefert die Bildidee, bestimmt die Form der Bildelemente, liefert die Bildordnung. Das ganze Kunstwerk ist aus der semantischen Kausalitätsverflechtung der gegenständlichen Darstellungsweise durch den Schneideakt der Schere herausgelöst.

Die Techniken der Zufallserzeugung erweitern sich stetig. Von der angesprochenen Schere über Würfel bis hin zum Computer, der den Zufall schließlich in der generativen Kunst einsetzt. Andererseits kritisiert Guderian in [15, S. 13] die Zufallsexperimente mit dem Computer in Bezug auf einen tatsächlichen Zufall, der auch bei der Vorgehensweise von Hans Arp nicht gegeben ist:

> Aber auch hier bleibt der Traum von der Darstellung eines totalen Zufalls nur ein Traum. Der Künstler führt Regie, indem er eine Auswahl aus der vorhandenen Materialfülle trifft.

Auch in der dadaistischen Literatur wurde vom Zufall Gebrauch gemacht. Hans Richter, selbst dadaistischer Maler, Graphiker, Kunstschriftsteller und Filmkünstler (vgl. [47]), schreibt dem Zufall in [20] eine wichtige Position im Dadaismus zu:

> Der Zufall wurde unser Markenzeichen. Wir folgten der Richtung, die er angab, wie einem Kompaß. Wir griffen damit in ein

Abbildung 2.10: Hans Arp: Collage nach den Gesetzen des Zufalls geordnet, Collage, 48,6 × 34,6 cm, 1917. Bildquelle: [6, S. 31].

> Gebiet über, von dem wir nichts oder wenig wußten, dem aber andere Persönlichkeiten sich in anderen Gebieten schon früher zugewandt hatten. Selbst in unseren Gesprächen und Diskussionen begann der Zufall eine Rolle zu spielen, in der Form einer mehr oder weniger assoziativen Sprechweise, in welcher uns Klänge und Formverbindungen zu Sprüngen verhalfen, die scheinbar Unzusammenhängendes plötzlich im Zusammenhang aufleuchten ließ.

Mathematiker und Naturwissenschaftler befassen sich vorwiegend mit dem berechenbaren Zufall. Dieser Bereich der Wissenschaft nutzt Wahrscheinlichkeitsrechnungen, um das Eintreten von Ereignissen zu kalkulieren. Der Ausgangspunkt sind also zufällige Ereignisse, deren Eintreten von einer Wahrscheinlichkeit abhängig ist.

Für die Erzeugung eines zufälligen Ereignisses oder zufälliger Daten ist in der generativen Computerkunst ein Zufallszahlengenerator vonnöten. Beispielsweise werden auf http://www.random.org Zufallszahlen generiert, die auf kosmischer Hintergrundstrahlung basieren und somit auf natürliche Art zufällig sind. Weitere Zufallszahlen, die auf Basis realer physikalischer Prozesse gewonnen werden sind z. B. der Zerfall radioaktiver Substanzen, thermisches Rauschen oder Elektronenspins. Für die Verwendung solcher natürlichen Zufallszahlen sind mechanische oder elektrische Module wie z. B. physikalische Rauschgeneratoren notwendig, die an den Rechner angeschlossen werden. Allerdings werden solche Zufallsquellen in der Regel nur selten verwendet, da sie eine niedrige Erzeugungsgeschwindigkeit besitzen und in ihrer Natur entsprechend nicht deterministisch, also nicht nachvollziehbar sind (vgl. [22, S. 354, 357]).

In vielen wissenschaftlichen oder künstlerischen Anwendungen ist diese Nachvollziehbarkeit allerdings sogar gewünscht, um z. B. Simulationen mehrfach unter den gleichen Bedingungen testen zu können. Dazu werden Zufallszahlen über eine Zufallsfunktion direkt im Computer erzeugt. Computer sind in ihrer Natur deterministisch. Sie arbeiten Algorithmen ab, die per Definition endlich sind. Das macht sie nachvollziehbar und rational erklärbar. Sie können durch diesen Determinismus keinen realen Zufall erzeugen, denn die dafür verwendeten Algorithmen sind ebenfalls deterministische Funktionen, die bestimmten Regeln folgen. Dennoch wird ein Zufall simuliert und man spricht deshalb auch von Pseudo-Zufallszahlen. Das bedeutet, dass die Zufallsfunktion bei gleichem Startwert immer auch das gleiche Ergebnis liefern wird. Die Ergebnisse liegen innerhalb eines definierten Wertebereichs und sie wiederholen sich, sobald die Periode des Zufallszahlengenerators erreicht ist. Einen guten Zufallszahlengenerator zeichnet aus, dass er in dem vorgegebenen Intervall eine möglichst gute Verteilung sowie eine möglichst große Periode erreicht (vgl. [17, S. 52] [15, S. 15] [22, S. 354, 357]).

In der Kunst kehrt sich die Methodik vom zufälligen Ereignis zur Wahrscheinlichkeit des Ereignisses für gewöhnlich um. Die Wahrscheinlichkeit wird vom Künstler vorgegeben und anschließend ein Zufallsversuch ausgeführt, der nach einer gewissen Anzahl von Ausführungen zu einem aleatorischen Kunstwerk führt (vgl. [15, S. 13]). Guderian listet für diese künstlerische Methodik in [15, S. 16f] noch zwei verschiedene Ausprägungen auf:

> Die einen [...] akzeptieren im Prinzip jedes in dieser Weise erhaltene Ergebnis, oder sie ändern die Rahmenbedingungen.
>
> Andere [...], lassen sich (vom Computer) zu einem System einmal festgelegter Rahmenbedingungen viele verschiedene Entwürfe vorschlagen und wählen daraus solche aus, die auch ihren eigenen künstlerisch-ästhetischen Ansprüchen genügen.

Abschließend stellt Guderian noch seine Katastrophentheorie und ihre Bedeutung für die aktuelle Kunst vor. Er definiert sie in [15, S. 23] wie folgt:

> Eine Katastrophe führt eine Konstellation (eine Großwetterlage, in einem Stausee gestautes Wasser, eine Stadt vor einem Bombenangriff, ein Brautpaar vor dem „Ja“-Wort in der Kirche) aus einem Gleichgewichtszustand irreversibel (unumkehrbar) in einen anderen Gleichgewichtszustand über.

Dabei muss eine Katastrophe, wie die Beispiele zeigen, nicht immer negativ behaftet sein. Ebenso können Katastrophen bewusst oder durch einen Zufall hervor gerufen werden. Allerdings schließt die Unumkehrbarkeit der Katastrophen laut Guderian aus, dass man sie selbst sieht. Stattdessen sind eher die Zustände vor oder nach der Katastrophe zu sehen. Guderian beschreibt in [15, S. 24], wie sich Künstler diese Zustände zu Nutze machen (vgl. [15, S. 23f]):

> Während die einen im Betrachter, der sich die Katastrophe „ausmalt“, erwartungsvolle Spannung hervorrufen, arbeiten die anderen mit dem Entsetzen oder Staunen über den nach der Katastrophe gegebenen Zustand.

In der generativen Kunst ist Zufall oder simulierter Zufall ein sehr wichtiges Werkzeug, um der Realität, die nicht perfekt ist, ein Stück näher zu kommen. Die Perfektion der Berechnungen von digitalen Maschinen kann mit Hilfe von Zufallszahlen abgeändert werden, sodass sie Abweichungen aufweist, die in menschlichen Augen natürlicher wirken, weil sie unserer analogen Welt ähnlicher sehen.

Ein gutes Beispiel für die menschliche Empfindung von Ästhetik ist die Symmetrie der Gesichtshälften. Wissenschaftliche Forschungen haben ergeben, dass Gesichter umso schöner empfunden werden, desto symmetrischer ihre Gesichtshälften sind. Im Gegenzug dazu werden perfekt symmetrische Gesichter, die mit Hilfe von Computern erzeugt wurden, entgegen der Erwartung, nicht schöner empfunden als leicht ungleiche Gesichtshälften (Abb. 2.11). Forscher der Universität Regensburg schreiben dazu in [33, S. 46f] folgendes:

> Die Ergebnisse aus unserem Experiment zur Symmetrie zeigen, dass Symmetrie zwar ein Faktor ist, der Attraktivität beeinflusst, jedoch bei weitem nicht in dem Ausmaß, wie es häufig behauptet wird. Es gilt vielmehr: Gesichter, die sehr asymmetrisch sind, sind eher unattraktiv [...], aber sehr unattraktive Gesichter sind deswegen noch lange nicht automatisch asymmetrisch [...]. Umgekehrt gilt ebenso: Sehr symmetrische Gesichter sind noch lange nicht attraktiv und sehr attraktive Gesichter zeigen durchaus Abweichungen von der Symmetrie.

Es gilt also, den Mittelweg zu finden und die Perfektion leicht abzuwandeln. Besonders für diesen Zweck findet der Zufall bzw. Pseudo-Zufall u. a. auch

(a) original **(b)** gespiegelt

Abbildung 2.11: Originales Selbstporträt im Vergleich zu einem „Chimärengesicht“ mit gespiegelter Gesichtshälfte (mit Ausnahme der Haare).

in der generativen Computerkunst Verwendung. Galanter sieht das in [45, S. 14] ebenso:

> In the era of computer-generated art the use of pseudo-random number generators becomes perhaps the most popular digital generative technique.

Im Gegensatz dazu warnt Golan Levin vor dem übermäßigen bzw. unüberlegten Einsatz von Zufall. In [56] macht er darauf aufmerksam, dass der Einsatz von Zufall allein noch nicht die generative Kunst macht:

> I'm not sure why today's digital artists are so drawn to aleatoric uses of randomness. [...] It's one thing to endorse the beauty of unexpected outcomes, but we must confront the fact that our algorithms are capable of coldness and ugliness, too, or we will never learn anything.

Kapitel 3

Techniken und Erscheinungsformen

Es können unterschiedliche Techniken genutzt werden, um generative Werke zu erzeugen, die sowohl in auditiver als auch visueller Form entstehen können. Da in dieser Thesis ausschließlich Computerkunst betrachtet wird, kann vorausgesetzt werden, dass mit diesen Techniken auch ausschließlich digitale Daten verarbeitet werden. Das bedeutet, dass auch die Anwendung der angeführten Techniken auf Foto-, Video- und Audiomaterial gleichermaßen möglich bzw. übertragbar ist. Beispielsweise kann digitales Rauschen auf Bildmaterial ebenso angewendet werden wie auf Tonmaterial.

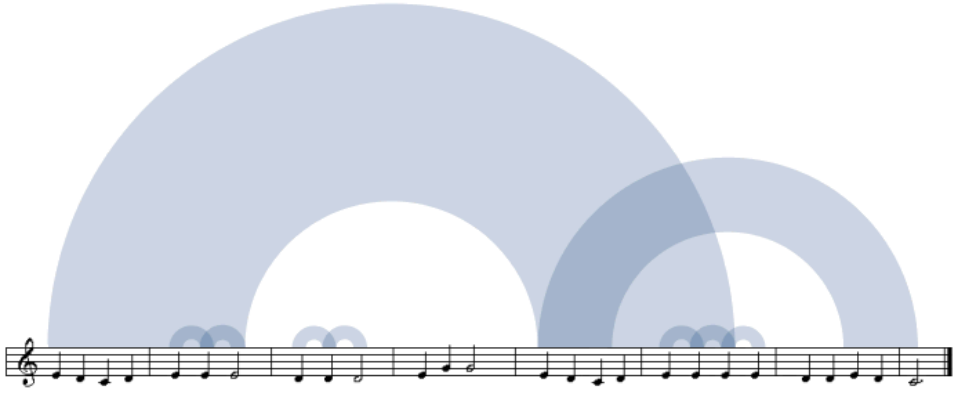
3.1 Wiederholung

In der Natur finden sich zahlreiche Beispiele für sich wiederholende Muster und Systeme. Das wohl größte natürliche System ist die Evolution. Jedes Lebewesen hat eine bestimmte Lebenserwartung, in der es die Möglichkeit hat, sich fortzupflanzen. Dadurch entstehen neue Generationen, die eine Wiederholung der vorigen Generation darstellen.

Allerdings sind Wiederholungen in der Realität selten exakt gleich. Vielmehr stellt jede Iteration einer Wiederholung einen Fortschritt in einem Prozess dar, welcher zu Variation führt. Durch Variation wird Abwechslung geschaffen, die man in der Evolution als Mutation bezeichnen würde. Eine leichte Variation lässt hierbei immer noch die Grundstruktur erkennen. Hingegen kann eine starke Variation ein ganz neues Gefüge produzieren, sodass die ursprünglich repetitive Quelle nicht mehr nachzuvollziehen ist.

Ebenso liegt die Wiederholung in der menschlichen Natur. In manchen Fällen können sich schnell wiederholende Blitzlichter sogar epileptische Anfälle auslösen. Ein allgemeineres Beispiel ist eine Lernmethode, bei der man sich Dinge einprägt indem man sie mehrfach wiederholt.

Auch in der Musik ist die Wiederholung eines der wichtigsten Instru-



(a) Bogendiagramm: „Mary Had a Little Lamb“.

(b) Bogendiagramm: Eine von Bachs Goldberg Variationen.

Abbildung 3.1: Martin Wattenberg: The Shape of Song, arc diagrams, 2001. Bildquelle: [73].

mente. Takte wiederholen sich in einem bestimmten System und Rhythmen bauen darauf auf. Es gibt Motive bzw. einprägsame Melodien, die innerhalb einer Komposition bewusst wiederholt werden. Besonders elektronische Musik macht sehr starken Gebrauch von Wiederholung. Im Unterschied zu klassischer Musik hat sie allerdings zusätzlich die Möglichkeit, mit Hilfe von Modulation zu variieren und ist daher nicht zwangsläufig auf die Variation der Melodie an sich angewiesen.

Martin Wartenberg analysiert in The Shape of Song (2001) Musikstücke in Bezug auf Wiederholungen und visualisiert sie in „arc diagrams“ (Abb. 3.1). Diese Diagramme bestehen aus Bögen, die wiederkehrende Muster in den Werken entlang einer Zeitleiste miteinander verbinden (vgl. [72, 73]).

Auch andere zeitbasierte Werke wie Videos und Animationen oder sogar Live Performances machen zahlreichen Gebrauch von Wiederholungen. Beispielsweise wurden in den Experimentalfilmen „Arnulf Rainer“ von Peter Kubelka und Tony Conrads „The Flicker“ ausschließlich schwarze und weiße Einzelbilder aneinander geschnitten. Die Wahrnehmung bezieht sich somit

Abbildung 3.2: Geodätische Kuppeln des Eden Projects in Cornwall, 2006. Bildquelle: [35].

nur noch auf den binären Status in schwarz oder weiß und den Rhythmus des Wechsels (vgl. [19, S. 49]).

Die Wiederholung ist wohl eine der wichtigsten Mechaniken und gleichzeitig auch Stärken des Computers. Er ist dafür geschaffen worden, Prozesse und Berechnungen immer wieder auszuführen. In der Computergrafik werden z. B. Texturen eingesetzt, die auf eine beliebige Größe immer wieder aneinander gereiht werden. Durch entsprechende Lichtsetzung können die simplen Wiederholungen dann wieder variiert werden. Auch die Animation von Massen wird durch den Einsatz von Computern möglich. In der Software Massive können beispielsweise individuelle Bewegungen auf viele verschiedene Akteure angewendet werden. Diese wiederholten Animationen werden dann zueinander versetzt, sodass der Eindruck entsteht, dass jedes Individuum eine eigene Aktion verfolgt (vgl. [19, S. 53, 55]).

Die Aufteilung in kleinere Module, die zu einem größeren Ganzen zusammengesetzt werden, deuten auf einen Teilaspekt der Wiederholung hin – die Modularität. Module sind gekapselte Elemente, die miteinander kombiniert werden können. Je modularer ein Aufbau ist, desto mehr Kombinationsmöglichkeiten gibt es. Module sind somit so geschaffen, dass sie möglichst allgemein und demnach auch wiederholt eingesetzt werden können. In der Programmierung sind beispielsweise Funktionen oder Prozeduren Module, die ein Teilproblem des gesamten Programms zum Teil wiederholt lösen. Ebenso wird Modularität häufig in der Produktion physischer Objekte benutzt um Kosten zu sparen. Teilweise sind diese Module sogar standardisiert und vorproduziert. Ausgereizt hat diese Methode Buckminster Fuller in den 50er Jahren mit seinen „geodesic dome designs“ (Abb. 3.2), die ausschließlich aus uniformen Elementen zusammengesetzt wurden (vgl. [19, S. 57]).

Eine Wiederholungstechnik ist das Erzeugen von Mustern. Muster bestehen in ihrer Natur aus Algorithmen. Selbst jahrhundertealte Muster folgen bestimmten Produktionsregeln, die in Quelltext übertragen werden können.

Eine weitere Wiederholungstechnik ist die Rekursion[1] (vgl. [19, S. 59, 63]). Sie ist eine besondere Art der Wiederholung, die durch einen Selbstaufruf entsteht. Eine rekursive Definition muss also sich selbst einschließen. In der Programmierung heißt das, die rekursive Funktion oder Prozedur muss sich selbst aufrufen, was zu ihrer eigenen Wiederholung führt. Daraus resultiert allerdings unweigerlich eine Endlosschleife und deshalb benötigt jeder rekursive Aufruf eine Abbruchbedingung. Diese Bedingung stellt sicher, dass der Algorithmus zu einem bestimmten Zeitpunkt ein Ende hat. Die Idee hinter Rekursion ist die Zerlegung eines Problems in immer kleinere Teilprobleme. Diese Teilprobleme sind ab einem gewissen Grad sehr leicht lösbar und können dann rückwirkend im rekursiven Aufstieg als Lösung der vorherigen Teilprobleme eingesetzt werden. Dieses Prinzip wird in der Informatik auch „teile und herrsche“ bzw. im Englischen „divide and conquer“ genannt. Anwendung findet es beispielsweise auch in Lindenmayer Systemen, die in Abschnitt 3.3.3 näher erläutert werden.

3.2 Transformation

Bei einer Transformation wird Material geformt oder vermischt, sodass sich ein neues Werk daraus ergibt. Ein Bildhauer beispielsweise schlägt bewusst Teile eines Steinblocks ab, was letztlich sein Kunstwerk hervorbringt. Er transformiert somit den Steinblock durch das Abschlagen in eine Skulptur. Auffällig ist hierbei, dass er für seinen Akt ein Ausgangsmaterial – den Steinblock – benötigt. Transformation wird also immer auf ein bereits existierendes Objekt angewandt und bringt etwas neues hervor (vgl. [19, S. 67]).

Die Veränderung findet vorerst nur am Objekt statt, z. B. in Form, Verhalten oder im Kontext. Ein anderer Kontext kann beispielsweise ein anderer Ausstellungsort eines Kunstwerks sein, welches unterschiedlich interpretiert wird, wenn man es im Kontext seiner Umgebung betrachtet. Aber ebenso führt die Transformation zu einer Veränderung der Beziehung zwischen Objekt und Betrachter. Transformation bzw. Konvertierung eines Farbfotos in ein schwarz-weißes Foto z. B. lässt das Bild emotionaler, zeitloser und nostalgischer wirken (vgl. [19, S. 69]). Ebenso kann durch die Wahl des Ausschnitts die Geschichte, die das Foto erzählt, beeinflusst werden, indem man bestimmte Dinge im Bild bewusst hervorhebt oder gar heraus nimmt.

In [19, S. 71–79] werden folgende Arten von Transformationen aufgeführt, die im Anschluss näher erläutert werden:

- Geometrische Transformation,
- Numerische Transformation,
- Transkodierung.

[1] s. Rekursion.

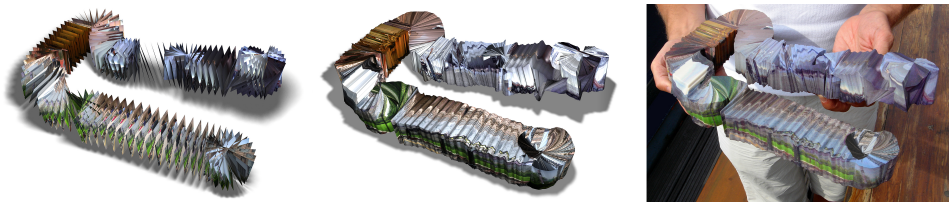


Abbildung 3.3: ART+COM: The Invisible Shape of Things Past, 1995–2007. Bildquelle: [29].

3.2.1 Geometrische Transformation

Die einfachste geometrische Transformation ist die Translation. Die Bewegung eines Objekts ist ebenso die fundamentalste aller Transformationen. Eine weitere Möglichkeit ist die Rotation um einen Mittelpunkt, der nicht zwangsläufig der Mittelpunkt des Objekts sein muss und sogar außerhalb liegen kann. Dieser Fakt wird allerdings erst in Bezug auf Animation relevant, da dieser Rotationsmittelpunkt erst in der Bewegung eines Objekts bedeutend wird. Während zweidimensionale Objekte nur um eine Achse gedreht werden können, besitzen dreidimensionale Objekte gleich drei Rotationsachsen. Die dritte Form der geometrischen Transformation ist die Skalierung, die in der Realität als Bewegung nur selten vorkommt, da sie auch eine Veränderung des Volumens voraussetzen würde. Im dreidimensionalen Raum allerdings kann der optische Eindruck einer Skalierung dennoch entstehen, wenn sich ein Objekt dem Betrachter nähert oder sich von ihm entfernt.

Außerdem ergeben sich durch die digitale Repräsentation von Objekten noch weitere geometrische Transformationsmöglichkeiten wie Spiegelung, Verzerrung, Dehnung, Scherung oder Verkrümmung (vgl. [19, S. 71]).

In „The Invisible Shape of Things Past“ (Abb. 3.3) wird die Transformation der Kamera eines Videos über die Analyse des Videomaterials errechnet. Die Einzelbilder des Videos werden anschließend entsprechend der berechneten Kamerabewegung im dreidimensionalen Raum platziert bzw. transformiert und ergeben so eine Skulptur, die Zeit konserviert. Hintergrund und Motivation dieses Projekts ist in [29] wie folgt beschrieben:

> Beeinflusst durch den aufkommenden Film und mehrfach belichtete Fotografien, lösten Kubisten und Futuristen in ihren Bildern und Skulpturen die lineare Darstellung von Raum und Zeit auf. Sie versuchten, Darstellungsformen für Bewegung zu finden und führten die Abbildung multipler Zeiten und Perspektiven eines Objekts ein.
>
> [...] Die Arbeit war davon motiviert, eine Gegenposition zu der damals verbreiteten Manie des Hyperrealismus in der Computergrafik zu manifestieren. Darüber hinaus ging es darum, eine

auf generativen Prozessen statt auf manuellen Modellierungen basierende architektonische und skulpturale Formfindung vorzustellen.

3.2.2 Numerische Transformation

Die Natur digitaler Objekte liegt in einer numerischen Form. Prinzipiell bestehen alle digitalen Daten vorerst aus Nullen und Einsen. Das ermöglicht demnach auch numerische Transformationen. Beispielsweise werden digitale Bilder durch Pixelwerte beschrieben, die mit Hilfe von mathematischen Formeln in Farbe bzw. Helligkeit verändert werden können. Diese Formeln können weitaus komplexere Veränderungen hervorrufen als bei geometrischen Transformationen. Ein gutes Beispiel sind Filterfunktionen, die ausschließlich die Pixelwerte eines Bildes, vorwiegend in Bezug auf Nachbarpixel, verändern und so z. B. folgende Operationen ermöglichen:

- Unschärfe,
- Schärfe,
- Kantendetektion,
- Farbkonvertierung.

Man unterscheidet grundlegend zwischen zwei Arten von Filtern: Hochpass- und Tiefpassfilter. Tiefpassfilter gleichen starke Veränderungen im Bild aus und führen so zu einem weicheren sowie unschärferen Bild. So kann beispielsweise Rauschen aus einem Bild entfernt werden. Hochpassfilter hingegen bewahren harte Übergänge oder verstärken sie gar. Das führt zu verstärkten Kanten und somit zu einem schärferen Bild (vgl. [19, S. 75]).

Ebenfalls ein Ergebnis numerischer Transformation ist ASCII Art. Hierbei werden die Farb- bzw. Grauwerte eines Bildes oder Videos in Zeichen transformiert. Jedes Zeichen aus der ASCII-Tabelle wird dazu für einen bestimmten Bereich von Farb- bzw. Grauwerten eingesetzt. Die ASCII-Tabelle wurde 1963 standardisiert und besteht aus insgesamt 128 Zeichen wovon 95 Zeichen druckbar sind. Die ältesten bekannten Werke, die dem Stil von ASCII Art gleich kommen, sind die Arbeiten von Computerkunstpionier Kenneth Knowlton. Zusammen mit Leon Harmon veröffentlichte er 1966 „Studies in Perception #1“ (Abb. 3.4), was allerdings nicht nur aus Zeichen der ASCII-Tabelle besteht (vgl. [31]).

3.2.3 Transkodierung

Prinzipiell bezeichnet Transkodierung vorerst nur die Konvertierung von digitalen Daten in ein anderes Format digitaler Daten. In der generativen Kunst werden aber Transformationen beispielsweise auch verwendet, um Daten eines bestimmten Bereichs, wie z. B. Temperaturen oder Börsendaten, einem komplett anderen Bereich, z. B. Farben oder Tönen, zuzuordnen und sie so-

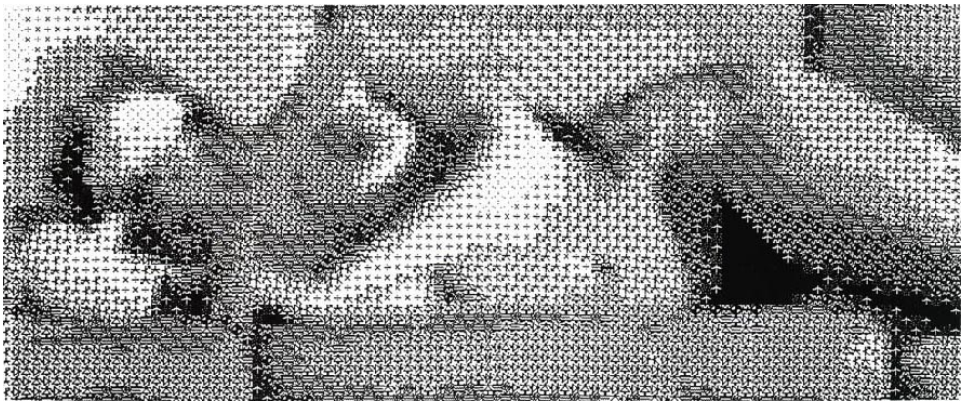
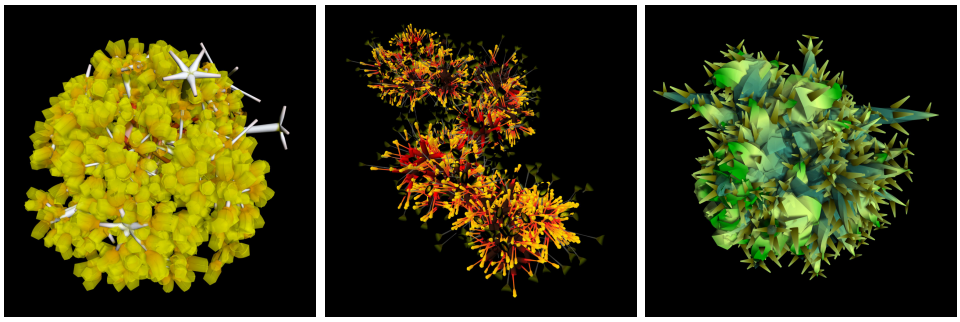


Abbildung 3.4: Ken Knowlton und Leon Harmon: Studies in Perception #1, 1966. Bildquelle: [51].



(a) IRCBot **(b)** PWS-Lineage **(c)** Virutmytob

Abbildung 3.5: Alex Dragulescu: Malwarez, 2007. Bildquellen: [68] [19, S. 80].

mit sicht- bzw. hörbar zu machen (vgl. [44, S. 14]). Diese übergreifende Verwendung der Daten ist dann ebenfalls eine Art der Transkodierung und bietet viele Möglichkeiten, komplett neue Beziehungen in den vorliegenden Daten herzustellen oder sie sogar in einen gänzlichen neuen Kontext zu setzen (vgl. [19, S. 79]).

Beispielsweise transkodierte Alex Dragulescu u. a. die Quelltexte von Computerviren mit Hilfe von Processing in 3D-Objekte, die ähnlich aussehen wie ihre biologischen Artgenossen (Abb. 3.5).

3.3 Simulation

Im Bereich der Simulation gibt es zahlreiche Anwendungsmöglichkeiten. Prinzipiell unterscheiden kann man die realistische und die artistische Simulati-

on. Das Ziel der realistischen Simulation ist die möglichst genaue Darstellung und Imitation der Wirklichkeit. Sie findet in vielen wissenschaftlichen Feldern Anwendung, wie z. B. bei Wettervorhersagen oder dem Katastrophenschutz. Auch die Filmindustrie bedient sich realistischer Simulationen, um beispielsweise digitales Feuer, Wasser oder Rauch zu generieren. Die detailgetreue Nachahmung der Realität ist in der artistischen Simulation jedoch nicht das Ziel, sondern das Ergebnis und die Erfahrung, die sie produziert und diese Erfahrung kann fernab jeglicher Realität liegen.

Auch in Spielen wird Simulation genutzt, um den Spieler in ein Szenario zu versetzen, dass nicht statisch ist, sondern sich den Aktionen des Spielers anpasst. Spieleentwickler haben so die Möglichkeit, ganze Welten zu schaffen, in denen der Spieler frei entscheiden und interagieren kann.

Beispielsweise wird in dem Rundenstrategie-Klassiker von 1996 „Civilization 2“ die ganze Erde mitsamt ihren Kulturen simuliert. Mittlerweile ist die Globalstrategie-Reihe des Entwicklers Sid Meier beim fünften Teil angekommen. Der Spieler kontrolliert die Wirtschaft, Technik und Bildung, das Militär und viele andere Aspekte seiner Zivilisation und muss so ihre Entwicklung vorantreiben. Er hat die Wahl, sich z. B. kriegerisch oder diplomatisch mit den anderen Zivilisationen auseinander zu setzen. Aufsehen erregte eine einzige Partie „Civilization 2“ von James Moore, an der er nunmehr zehn Jahre spielt. Neben der Fachpresse berichteten auch die klassischen Medien in Radio, Fernsehen und Tagespresse über seinen Erfahrungsbericht, den er am 12. Juni 2012 auf der Social-News-Plattform Reddit unter dem Spitznamen „Lycerius“ veröffentlichte (s. auch [62]). Moore begann die Partie im Jahre 2002 mit den Kelten, befindet sich mittlerweile im Jahr 3991 und es herrscht seit 1700 Jahren Krieg zwischen seiner Nation und den zwei noch verbliebenen vom Computer gesteuerten Völkern. Nuklearschläge sind an der Tagesordnung. Sie führen zu einer hohen Verseuchung und Umweltverschmutzung, die die Polkappen schon 20 Mal zum Schmelzen gebracht haben. Die globale Erwärmung zerstört die fruchtbaren Böden und verwandelt sie in Dschungel oder Einöden. Durch Überbevölkerung und zu wenig Landwirtschaft verhungern die Menschen und Städte schrumpfen oder verschwinden gänzlich von der Landkarte, weil alle Investitionen auf den Krieg ausgerichtet sind. Dieses fatale Zukunftsszenario basiert zwar nicht auf einer wissenschaftlichen Simulation, dennoch ist es keine Utopie und auch ein Zeichen dafür, dass Simulationen wach rütteln und Aufmerksamkeit für Punkte schaffen können, die vorher evtl. nicht oder nur in geringem Maße bedacht wurden (vgl. [39]).

Ebenso spielen Physiksimulationen in Spielen, aber auch in der Architektur und im Maschinenbau eine entscheidende Rolle. Durch sie ist es möglich, Kräfte und Massen in einer virtuellen Umgebung zu testen und daraufhin das Modell optimal in Struktur und Material herzustellen. So kann z. B. die Statik eines Gebäudes während eines Erdbebens bereits vor dem Bau virtuell getestet und daraufhin strukturschwache Stellen verstärkt werden (vgl. [19,

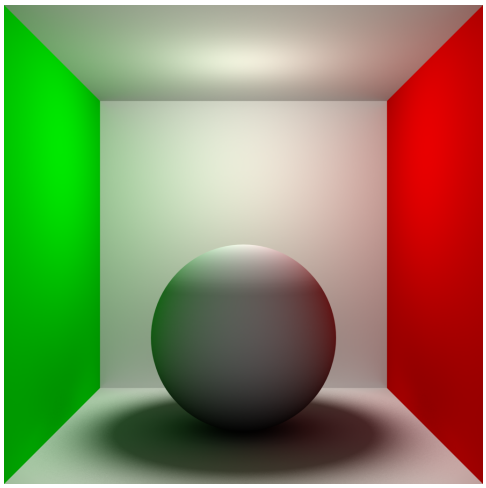


Abbildung 3.6: Color Bleeding: 3D-Rendering-Effekt, bei dem Farbe zwischen nah beieinander liegenden Objekten übertragen wird.

S. 151]).

Auch für fotorealistische 3D-Renderings werden Simulationen eingesetzt. Beispielsweise werden für die Berechnung der Beleuchtungsverhältnisse virtuelle Photonen von den Lichtquellen in den Raum geschossen. Dort kollidieren sie mit Objekten und beleuchten diese. Außerdem interagieren sie mit dem Material der Objekte, nehmen deren Farbe auf („Color Bleeding“, vgl. [32], Abb. 3.6) und werden auf andere Objekte im Raum reflektiert, wo sie wiederum interagieren. Eine perfekte Berechnung der Verteilung dieser virtuellen Photonen ist jedoch nicht möglich, da sie eine unendliche Anzahl an Photonen erfordern würde. Deshalb werden Zufallsfunktionen eingesetzt, um sie möglichst gleichmäßig im Raum zu verteilen. Auch diese physikalische Simulation weist somit generative Elemente auf.

Jede Simulation besteht aus drei Teilen: Variablen, einem System und einem Status. Eine Variable ist ein Wert, der einen Teil der Simulation darstellt. Das System sind die Regeln, nach denen sich die Simulation verhält und der Status des Systems sind die Werte der Variablen zu einer bestimmten Zeit. Soll nun beispielsweise ein fahrendes Auto simuliert werden, gilt es zunächst, die für die Simulation wichtigen Variablen zu finden. Die potentiellen Möglichkeiten an Variablen sind jedoch unbegrenzt. Wichtig ist somit, eine begrenzte Anzahl der tatsächlich relevanten Variablen zu definieren. Im Falle einer einfachen Simulation des Autos sind das z. B.: Größe, Geschwindigkeit, Fahrtrichtung und eine Farbe für die Visualisierung. Anschließend müssen die Regeln festgelegt werden, die das Verhalten des Autos im System bestimmen. Es kann beispielsweise beschleunigen, bremsen und in einer Richtung lenken. Geht man von einer komplexeren Simulation mit mehreren Autos aus, wären reaktive Regeln wie das Abbremsen vor einem Hindernis

zusätzlich nötig. Außerdem kann man eine Welt mit einbeziehen, in der sich die Autos bewegen. Eine Welt aus Straßen und Ampeln, die das System um weitere Variablen und Regeln erweitern. So entsteht eine immer komplexere Simulation, deren Ausgang immer ungewisser wird (vgl. [19, S. 149]).

Die Ungewissheit über den Ausgang einer Simulation ist ein wesentlicher Punkt. Eine Simulation und ihr System erschaffen zahlreiche Möglichkeiten von Formen, die in der Regel unvorhersehbar sind. Vergleichbar ist diese Situation mit der Berechnung einer bestimmten Stelle von Pi. Man muss mit der Berechnung von vorn beginnen und sich bis zur gewünschten Stelle vorarbeiten. Der Unterschied ist nur, dass die bereits berechneten Stellen von Pi statisch sind und sich nicht verändern. Bei einer Simulation hingegen kann durch die Veränderung der Variablen selbst die Ausgangssituation dynamisch gestaltet werden, sodass eine neue Berechnung ins Ungewisse gestartet werden kann.

3.3.1 Zelluläre Automaten

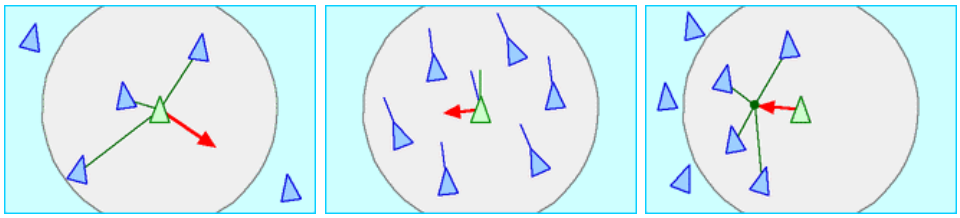
Ein zellulärer Automat ist ein in der Regel ein- oder zweidimensionales Feld bzw. Gitter, das mit Zellen besetzt ist. Er kann allerdings auch mehr als zwei Dimensionen besitzen. Die Zellen können bestimmte Status bzw. Zustände annehmen. Diese reichen von binären Zuständen wie an-aus oder schwarzweiß bis zu komplexeren Status mit mehreren Zuständen. Außerdem verhalten sich die Zellen nach bestimmten Regeln, die für alle Zellen gleichermaßen gelten. Diese Regeln verändern den Status der Zellen über die Zeit bzw. in eine folgende Generation in Abhängigkeit von den benachbarten Zellen. Die Zustände der Nachbarzellen bestimmen also, welche Regel auf eine Zelle angewandt wird (vgl. [22, S. 441]).

Der wohl bekannteste zelluläre Automat ist John Conway's Game of Life. Dieses Simulationsspiel wurde in Abschnitt 2.2 bereits näher erläutert (vgl. [19, S. 165] [77]).

3.3.2 Schwarmverhalten

Bei der Simulation von Schwärmen werden Agenten erzeugt, die einer kleinen Auswahl von Regeln folgen und sich so in einer gegebenen Umgebung bewegen. Diese Agenten können z. B. Fische, Vögel, Insekten oder Menschen sein, die sich in einer Menge, Herde oder einem Schwarm entsprechend ihrer Regeln verhalten. Diese Regeln beziehen sich auf Eigenschaften bzw. Status der Agenten sowie ihre lokale Umgebung. Beispielsweise folgt in dem Programm „Boids“ von Craig Reynolds aus dem Jahr 1986 jeder Agent drei folgenden Regeln, die die Basis vieler Schwarmalgorithmen darstellen (Abb. 3.7):

1. Lenke, um lokale Nachbarn nicht zu bedrängen.
2. Lenke in die durchschnittliche Richtung der lokalen Nachbarn.
3. Lenke zu der durchschnittlichen Position der lokalen Nachbarn.



(a) Trennung **(b)** Angleichung **(c)** Kohäsion

Abbildung 3.7: Basisregeln vieler Schwarmalgorithmen. Bildquelle: [66].

Schwarmalgorithmen werden aber beispielsweise auch eingesetzt, um die Entwicklung von Städten zu simulieren. So erhofft man sich, dynamische Pläne entwickeln zu können, die leichter verändert werden können und besser den Bedürfnissen der Bewohner angepasst sind (vgl. [19, S. 167] [66], s. auch [22, S. 398f]).

3.3.3 Lindenmayer Systeme

L-Systeme wurden 1968 vom ungarischen theoretischen Biologen Aristid Lindenmayer vorgeschlagen, um biologische Entwicklungen wie z. B. Pflanzenwachstum zu simulieren. Ihre Einsatzmöglichkeiten sind jedoch weitaus vielfältiger und erstrecken sich beispielsweise auch über die Erzeugung von Fraktalen in der Computergrafik. Ein L-System ist eine spezielle Grammatik zur Erzeugung von Zeichenfolgen, dessen Wörter grafisch interpretiert werden können. Das geschieht in Form von gezeichneten Linien (vgl. [58] [22, S. 468]).

Um L-Systeme zu beschreiben, wird in der Regel die in [22, S. 468f] aufgeführte Syntax verwendet:

Zeichnen von Strecken und Drehungen

F Zeichnen einer Linie bestimmter Länge (Forward)

α Drehwinkel

M Vorwärtsbewegung ohne zu zeichnen (Move)

$+$ Drehung um α entgegen dem Uhrzeigersinn

s Streckungsfaktor

$-$ Drehung um α im Uhrzeigersinn

$*$ die Linienlänge wird mit s multipliziert

$|$ Drehung um 180°

: die Linienlänge wird durch s dividiert

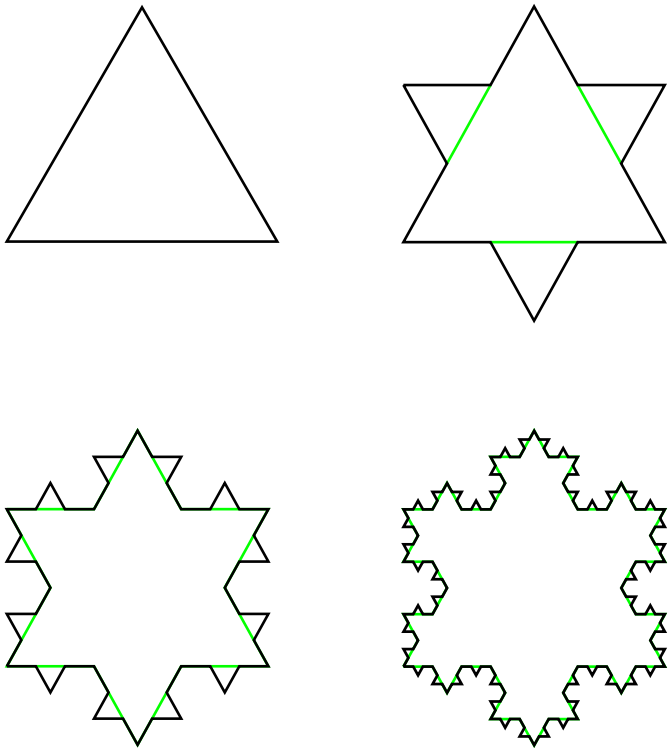


Abbildung 3.8: Die ersten vier Iterationen einer Koch-Kurve. Bildquelle: [52].

Stack-Operationen

[aktuellen Zustand (Position und Richtung) auf dem Stack speichern

] gespeicherten Zustand vom Stack holen

Ein historisch bedeutsames Beispiel ist die Koch-Kurve, auch Schneeflocken-kurve (Abb. 3.8) genannt. Sie wurde 1904 vom schwedischen Mathematiker Helge von Koch vorgestellt und lässt sich sehr einfach durch ein L-System erzeugen, das wie folgt definiert ist (vgl. [52] [22, S. 468]):

Axiom: $F++F++F$

Produktionsregeln: $F \rightarrow F-F++F-F$

3.3.4 Evolutionäre Systeme

Mit Hilfe von genetischen Algorithmen kann Evolution simuliert werden indem künstliche Gene erzeugt werden. Diese Gene können über Kreuzung oder Mutation verändert werden. Kreuzung ist dabei die Kombination der Gene zweier Eltern zu einem einzigartigen Kind. Es werden Teile der Gene beider Eltern miteinander kombiniert, sodass ein Kind mit der Mischung aus beiden entsteht. Bei einer Mutation wird das Gen einer einzelnen Kreatur verändert, welches dann weitervererbt wird. In der Natur ergeben sich Mutationen z. B. aus Kopierfehlern oder durch radioaktive Strahlung. In der Simulation von künstlichem Leben wird das durch die Veränderung von einem oder mehreren Zeichen eines Gens implementiert (vgl. [19, S. 167]).

Georg Trogemann definiert in [22, S. 489] folgenden genetischen Basis-Algorithmus:

1. Wähle eine geeignete Codierung des Genotypen.
2. Initialisiere eine zufällige Population von Genotypen.
3. Wiederhole bis die angestrebte Fitness erreicht ist, d. h. überprüfe, ob die Leistung der Phänotypen zufriedenstellend ist.
   (a) Bewerte alle Individuen der aktuellen Generation gemäß Fitness-Funktion.
   (b) Selektiere Paare gemäß Heirats-Schema und erzeuge Nachkommen durch Mutation und sexuelle Rekombination (crossover).
   (c) Wähle die ‚Besten' aus (survival of the fittest) und ersetze die aktuelle Generation durch diese Nachkommen.
   (d) Aktualisiere die Abbruchbedingung.

Genetische Algorithmen sind Teil der Simulation künstlichen Lebens. Künstliches Leben unterscheidet sich von künstlicher Intelligenz, indem es sich an Stelle von höheren kognitiven Funktionen auf die Imitation biologischer Phänomene konzentriert, um Reflexe, Verhalten und Evolution zu simulieren (vgl. [19, S. 167]).

Der Vorteil der Simulation gegenüber der biologischen Evolution ist die Möglichkeit, viel mehr Generationen in wesentlich kürzerer Zeit zu erzeugen. In Bezug auf generative Kunst können evolutionäre Simulationen der Erzeugung neuer Formen dienen indem diese förmlich „gezüchtet" werden. Auch als Problemlösungsmethode, vor allem in Bezug auf Optimierung von Entwürfen, werden genetische Algorithmen eingesetzt. Sie haben den Vorteil, dass sie zumeist gänzlich andere Formen generieren als Experten sie von Hand entwickeln. Außerdem ergeben sich daraus evtl. Formen, die traditionelle Entwurfsmethoden von vornherein nicht in Erwägung ziehen. Auch in der Architektur werden deshalb genetische Algorithmen eingesetzt, um neue innovative Formen zu finden (vgl. [19, S. 167f]).

3.4 Parametrisierung

Diese Methode bietet zahlreiche Kombinationsmöglichkeiten für den Gestalter. Er definiert nun nicht mehr nur noch ein einziges, finales Objekt, sondern schafft ein ganzes Konvolut von möglichen Werken. Im Allgemeinen kann man Parameter als einen Wert definieren, der das Ergebnis eines Prozesses beeinflusst. Es kann demnach alles ein Parameter sein, was einen Wert besitzt. Beispielsweise besitzt ein Auto eine bestimmte Anzahl von Türen, eine Farbe, eine Höchstgeschwindigkeit und während es fährt eine aktuelle Geschwindigkeit. Diese Parameter können statisch bzw. konstant oder dynamisch sein. Im Falle des Autos ist die aktuelle Geschwindigkeit dynamisch während sich die Höchstgeschwindigkeit für gewöhnlich nicht ändert.

Der Vorgang, diese variablen Elemente eines Prozesses zu finden und zu beschreiben, nennt man Parametrisierung. Die Aufgabe des Gestalters ist in diesem Fall, Grenzen zu definieren und die entstehenden Variationen in eine Richtung zu lenken. Je mehr Parameter er definiert, desto größer sind die Kombinationsmöglichkeiten unter den Parametern und somit auch die Anzahl der möglichen Ergebnisse.

Dennoch funktioniert Parametrisierung in der generativen und interaktiven Kunst nur in Kombination mit den übrigen Techniken. Es hat also keinen Effekt, wenn Parameter integriert sind, aber nicht in einer Wiederholung, Transformation oder Simulation eingesetzt werden, da die Parameter selbst per Definition nicht für die Änderung sorgen. Sie sind ausschließlich die Variablen im System, die Werte speichern und den entsprechenden generativen Techniken diese Werte zur Verfügung stellen. Parametrisierung stellt somit eine Brücke zwischen den verschiedenen vorgestellten Techniken dar (vgl. [19, S. 95]).

Einer der wichtigsten Aspekte der Parametrisierung ist die Kontrolle. Mit Hilfe von Parametern ist es den Gestaltern möglich, ihre Werke dynamisch oder gar interaktiv zu modifizieren. Dies kann beispielsweise durch den Einsatz von Zufallszahlen realisiert werden, die innerhalb bestimmter Grenzen verschiedenste Variationen liefern. Die Ergebnisse dieser Brute-Force-Methode (Lösungsmethode für Probleme aus den Bereichen Informatik, Kryptologie und Spieltheorie, die auf dem Ausprobieren aller oder zumindest vieler möglichen Fälle beruht [34].) liefern zwar eine hohe Quantität, aber in den seltensten Fällen ebenso Qualität. Daher sollten die quantitativen Ergebnisse im Anschluss erneut nach den qualitativen Kriterien des Gestalters gefiltert werden. Die umgekehrte Methode, die direkt mit den qualitativen Kriterien des Gestalters beginnt, ist ein System aus manuell vom Gestalter kontrollierten Parametern, die er so lange modifiziert, bis er sein gewünschtes Ergebnis bzw. genügend Variationen erreicht hat.

Ebenso besteht durch Parameter die Möglichkeit, diese untereinander zu verbinden und sie so in Beziehung zueinander zu setzen. Diese Beziehungen

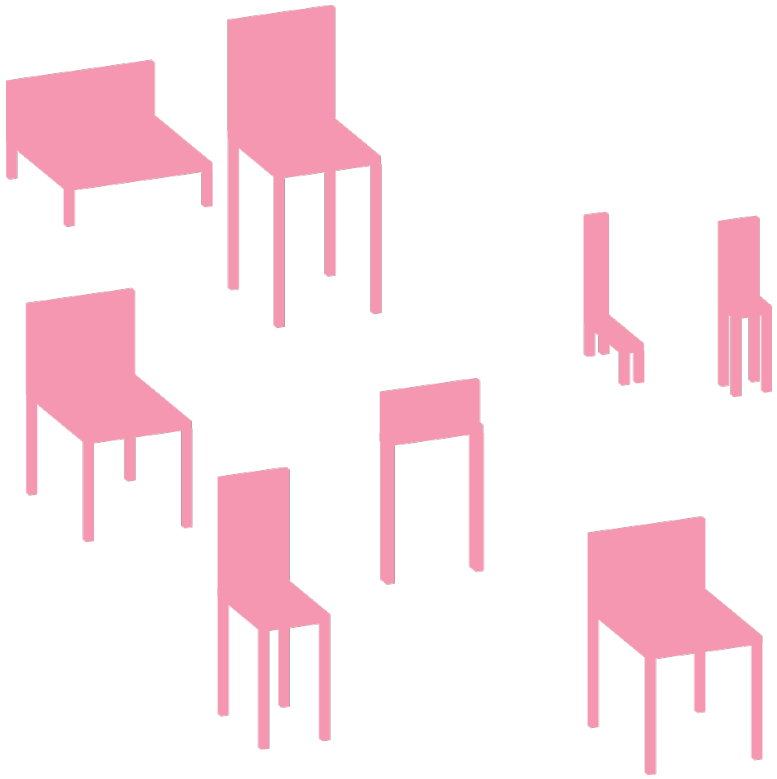


Abbildung 3.9: Variationen eines Stuhls durch Parametrisierung. Bildquelle: [19, S. 116].

oder Constraints[2] erzeugen dann ein dynamisches System, welches sogar auf indirekte Änderungen reagiert. Beispielsweise kann der Designer eines Stuhls in seinem digitalen Konzept die Stuhlbeine mit den Ecken der Sitzfläche verknüpfen. Sollte sich nun die Sitzfläche in Höhe, Tiefe oder Breite verändern, würden sich die Stuhlbeine durch die Verknüpfung dynamisch an die neuen Dimensionen anpassen. Die direkte Änderung der Sitzfläche impliziert demnach die indirekte Korrektur der Stuhlbeine (vgl. [19, S. 107]).

Außerdem könnte der Designer durch weitere Parametrisierung ebenso die Stuhlbeine direkt modifizieren und beispielsweise auch die Form und Größe der Lehne beeinflussen. Angenommen, er verbreitert dann die Sitzfläche und Lehne auf zwei Meter und kürzt die Stuhlbeine auf etwa 30 Zentimeter, wäre das Ergebnis eher ein Bett als ein Stuhl. Die Parametrisierung ermöglicht diese Variation, ohne dass ein zusätzlicher Aufwand nötig ist (Abb. 3.9).

[2] engl., Zwang, Auflage, Nebenbedingung.

Kapitel 4

Der Quellcode als Medium zwischen Künstler, Werk und Rezipient

In diesem Kapitel wird zunächst die klassische Dreiecksbeziehung zwischen Künstler, Werk und Rezipient (Abb. 4.1) aus Sicht der generativen und interaktiven Kunst betrachtet. Anschließend folgen Ansichten zur Bedeutung der Begriffe Medium und Code. Der letzte Abschnitt schlussfolgert daraufhin, inwieweit der Quellcode als Medium in dem klassischen Dreieck fungiert.

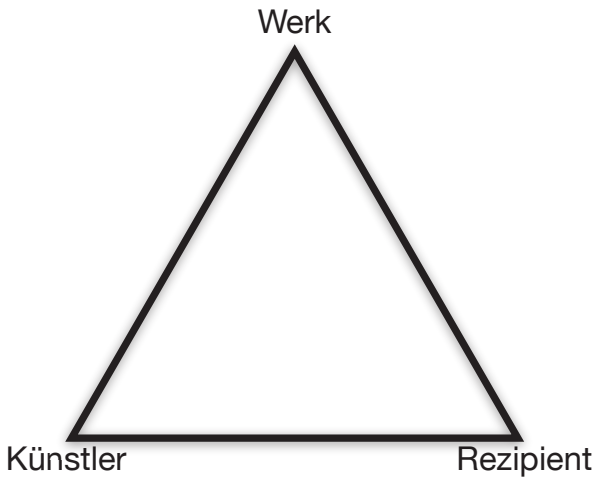


Abbildung 4.1: Die klassische Beziehung zwischen Künstler, Werk und Rezipient.

4.1 Die klassische Beziehung

4.1.1 Künstler

Georg Trogemann betrachtet den Künstler in Bezug zur Computerkunst und bezeichnet ihn deshalb auch als Autor, der interaktive Anwendungen schreibt. Seine Erwartungen an den Autor formuliert er in [22, S. 142] folgendermaßen:

> Der Autor muss dabei nicht nur in der Lage sein, die Prozesse und Abläufe, die entstehen, wenn sein Programm auf einem Rechner ausgeführt und auf einem Display zur Darstellung gebracht wird, bis zu einem bestimmten Grad zu antizipieren, sondern sie auch dramaturgisch und ästhetisch inszenieren. Die besondere Herausforderung künstlerischer Computeranwendungen besteht darin, dass im besonderen Maße Verstand und Sinnlichkeit, Gefühl und Vernunft verschmolzen werden müssen, und das auf der Basis einer bisher hauptsächlich der rationalistischen Tradition verpflichteten Maschine.

Auch Marius Watz beschreibt in [74] diesen Abstraktionsprozess:

> This is not to say that the use of parametric systems is trivial or simply a matter of powerful hardware. The process of abstracting aesthetic processes into executable computer code requires formal stringency and a talent for reverse-engineering a desired result in order to identify its causal elements. Every aspect of the system must be explicitly described, including details that might seem insignificant when considered individually. When considered as a whole, however, these series of decisions become the very body of the work.

Der Künstler muss also zunächst die zu kommunizierende Idee abstrahieren und sie in Quelltext als eine für die Maschine verständliche Sprache übersetzen. Um seine Inszenierung nicht zu zerstören, muss er Grenzen für die Interaktion mit dem Rezipienten definieren. Der Autor muss dazu die Handlungen des Rezipienten vorhersehen, was Trogemann als antizipieren bezeichnet. Das unterscheidet den Autor allerdings noch nicht vom Programmierer. Er muss außerdem stets die Ästhetik und Dramaturgie seiner Inszenierung im Blick haben und in der Lage sein, dies auch über die rationale Maschine zu transportieren (vgl. [22, S. 142]).

Bis es dem Gestalter gelingt, Softwarekunst zu veröffentlichen, durchläuft diese einen Prozess bis zu ihrer Fertigstellung (Abb. 4.2). Dieser Prozess beginnt mit der ursprünglichen Idee, die durch die bereits erwähnte Abstraktion vom Gestalter bzw. Autor in einen Algorithmus gewandelt wird. Der Algorithmus bestimmt das Wesen des Werks und die Regeln, nach denen es sich

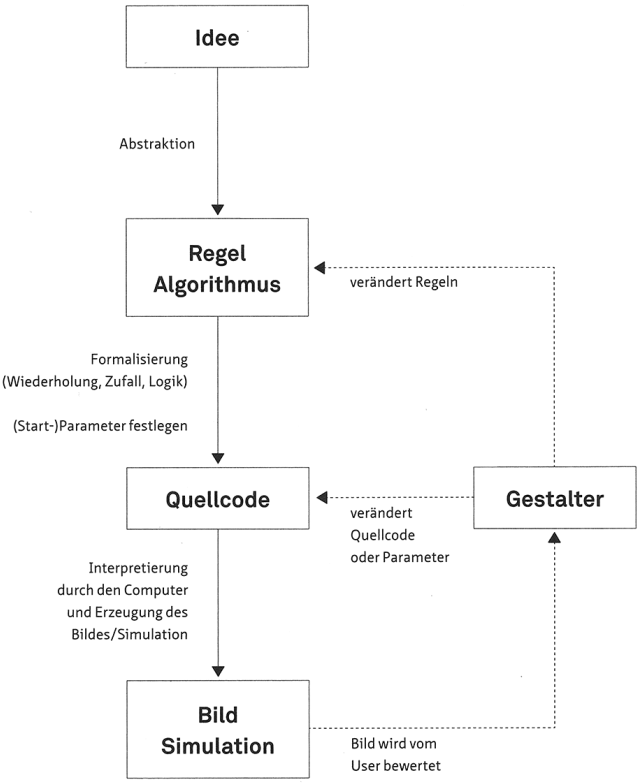


Abbildung 4.2: Gestaltungsprozess generativer Kunst. Bildquelle: [4, S. 461].

verhält. Allerdings ist selbst dieser Algorithmus nur ein formales Konzept solange er nicht in einen bestimmten Quellcode einer bestimmen Programmiersprache überführt wird. Erst dann kann er von einem Rechner ausgeführt und vom Benutzer erlebt werden. Der Gestalter erhält durch die Rezeption der Benutzer dann eine Resonanz, die ihn wiederum dazu veranlassen kann, das Werk zu beeinflussen.

Georg Trogemann sieht auch im Verhältnis zwischen Autor und Werk durch „die Eigenaktivität computerbasierter Medienobjekte neue Formen der Autorschaft entstehen.“ [22, S. 144] Er unterteilt die Ablösung des Werks vom Autor in folgende drei Stufen:

1. Die niedrigste Stufe besteht aus Systemvariationen und Wahlmöglichkeiten, die Trogemann als Hyperstrukturen bezeichnet. Der Autor bzw. Künstler definiert hier jedoch noch die Grenzen dieser Wahlmöglichkeiten, die er, wie zuvor bereits beschrieben, schon bei der Implementie-

rung antizipiert. Trogemann zählt diese Stufe genauso wie den Einsatz von aleatorischen, also zufälligen Ereignissen zum herkömmlichen Verhältnis zwischen Autor und Werk (vgl. [22, S. 144f]).

2. In der zweiten Stufe hat das System ein Gedächtnis, das durch Variablen implementiert wird. Außerdem kann das System mehrere Zustände einnehmen, die sich sowohl aus den Benutzereingaben als auch aus den internen Variablen ergeben. Somit können selbst gleiche Benutzereingaben zu unterschiedlichen Zeitpunkten durch das Gedächtnis des Systems unterschiedliche Ergebnisse liefern. Für den Benutzer, der die innere Funktionsweise des Systems nicht kennt, sind diese Ergebnisse nicht determinierbar. Dieses zeitabhängige Verhalten durch ein Gedächtnis veranlasst Trogemann, solche Systeme höher einzustufen (vgl. [22, S. 145]).
3. Die höchste Stufe stellt autonome Maschinen dar, die „in Form von lernenden Systemen, genetischen Algorithmen und selbstreproduzierenden Automaten“ [22, S. 144f] realisiert sind. Trogemann bezeichnet diese Lern- und Selbständerungsvorgänge als „Code-Reflexivitäten“, in denen das System „seine eigene innere Struktur im Laufe der Interaktion umbauen und selbständig weiterentwickeln“ [22, S. 144f] wird.

Diese drei Stufen zeigen, dass die Grenze zwischen Autor und Werk mit der zunehmenden Autonomie der Werke stärker verschwimmt. Weitere Ausführungen dazu folgen im Abschnitt 4.4.

4.1.2 Werk

Karlheinz Essl definiert in [42] das Kunstwerk als unantastbar:

> Mächtig und monolithisch steht es da, unerschütterlich, von eindrucksvoller Größe und Gehabe. In seiner Geschlossenheit und Vollkommenheit ist es einem Gesetzestext (sowohl im juristischen als auch im religiösen Sinne) vergleichbar, wie z. B. die Thora: diese ist zwar offen für vielfältige Interpretationen, darf aber selbst nicht verändert oder in Frage gestellt zu werden. Der Buchstabe ist heilig.

Im Anschluss macht er ebenfalls in [42] auf die Antithese des Werkes – den Prozess – aufmerksam:

> Die zeitliche Ausdehnung eines PROZESSES ist zumeist unbestimmt (im Unterschied zum exakt determinierten WERK). [...] Während sich WERKE dank ihrer Geschlossenheit und textuellen Eindeutigkeit hervorragend reproduzieren lassen (was dem bürgerlichen Konzertbetrieb entgegenkommt), verweigern sich PROZESSE dieser Verdinglichung: sie erscheinen einmalig und jedes mal anders, neu.

Tabelle 4.1: Hauptmerkmale von Werk und Prozess nach Essl. Quelle: [42].

Werk	**Prozeß**
geschaffen	verursacht
gestaltet	generiert
geschlossen	offen
endlich	unendlich
reproduzierbar	irreproduzibel
ziel-orientiert	weg-orientiert

Die Hauptmerkmale von Werk und Prozess fasst Essl in einer Gegenüberstellung zusammen (Tab. 4.1). Allerdings betont er, dass es seiner Meinung nach weder ein reines Werk noch einen reinen Prozess gibt. In der Realität handele es sich stets um eine Konvergenz der beiden Begriffe. Seine Argumentation mündet in einem Plädoyer für eine neue Sichtweise, „die WERK und PROZESS nicht als Widersacher gegeneinander ausspielt, sondern die Polarität dieser beiden Wesensformen in sich vereint.“ [42] Dazu führt er den von Umberto Eco Anfang der 60er Jahre geprägten Begriff des „Offenen Kunstwerkes“ fort, welcher in [5, S. 57] wie folgt definiert wird:

1. Kunstwerke in Bewegung - gekennzeichnet durch die Einladung, gemeinsam mit ihrem Hervorbringer das Werk zu machen;
2. für ständige Neuanknüpfungen von inneren Beziehungen, die der Rezipierende im Akt der Perzeption [...] entdecken und auswählen soll;
3. jedes Kunstwerk, auch wenn es nach einer ausdrücklichen oder unausdrücklichen Poetik der Notwendigkeit produziert wurde, wesensmäßig aber offen sind für eine virtuell unendliche Reihe möglicher Lesarten, deren jede das Werk gemäß einer persönlichen Perspektive, Geschmacksrichtung, Ausführung neu belebt.

Eco bezieht sich hierbei auf geschlossene Werke, „die durch ihre Komplexität vielfältige Lesarten erlauben und so den Betrachter zum Mitschöpfer machen“ [42]. Essl wendet diese Definition ebenso auf offene Prozesse an und stellt sich ein „Offenes Kunstwerk“ vor, das „den Rezipienten zu eigenen Erkundungsmärschen einlädt.“ [42] Der Rezipient kann sich im Kunstwerk also frei bewegen, wird aber durch Wegweiser als Orientierungshilfe gestützt. Das führt zu einer Vieldeutigkeit des Kunstwerks, das Essl mit der Struktur eines Hypertexts vergleicht. Der Rezipient nimmt verschiedene Vorgänge gleichzeitig war, bezieht sie aufeinander und interpretiert sie. „Erst im aktiven Vorgang des Betrachtens entsteht so – in einem zeitlichen Abtastvorgang – eine persönliche, erlebte ‚Fassung‘ des Werkes.“ [42]

4.1.3 Rezipient

Wolfgang Kemp sieht die Kunstbetrachtung als eine gemeinsame Aufgabe der Gesellschaft, bei der die Sinnsuche in der Rezeption selbst und nicht in der Botschaft des Werks, Autors oder der Meinung von Experten liegt. Für Kemp steht also der Rezipient im Vordergrund, dessen Aufgabe es ist, das Werk zu interpretieren. Dabei soll er sich nicht von der Intention des Autors oder den Interpretationen von Experten verleiten lassen. Kemp kritisiert in [13, S. 15]:

> Die Frage: „Was wollte der Künstler damit sagen?“ ist das Amen der ästhetischen Kirche

Diese Suche nach der Absicht wird auch als Intentionalismus bezeichnet. Doch ihre Bedeutung schwindet, da der Betrachter zunehmend an Macht gewinnt. Die Suche nach der Intention macht der Nachfrage des Rezipienten Platz. Vergleichbar ist diese Situation z. B. mit der künftigen Wandlung des Fernsehens durch die zunehmende Bedeutung des Internets. Fernsehsender bieten bereits heute Videos ihrer Sendungen auf ihren Webseiten an, die jederzeit vom Zuschauer abgerufen werden können und unabhängig von der Sendezeit sind. Es gibt bei dieser „Video on Demand“- oder „IPTV“-Technologie also keine zentrale Sendung für alle Zuschauer gleichermaßen zu einer bestimmten Zeit, sondern alle möglichen Sendungen zu einer beliebigen Zeit individuell für jeden Rezipienten. Der Internetnutzer entscheidet also, wann er welche Sendung sehen möchte.

Der Betrachter wird außerdem zunehmend selbst zum Schöpfer. Über zahlreiche soziale Plattformen im Internet kann er Texte, Fotos und Videos veröffentlichen während er gleichzeitig die Inhalte von anderen Teilnehmern des sozialen Netzwerks konsumiert. Er ist also Rezipient und Autor gleichermaßen.

Die Interaktion zwischen Rezipient und Werk bezeichnet Georg Trogemann bei computerbasierten Arbeiten als „Improvisation“, die eine Performance des Benutzers darstellt (vgl. [22, S. 142]). Er bezieht sich dabei ebenfalls auf das von Eco eingeführte „Offene Kunstwerk“ und stellt die Frage, inwieweit formale Programmstrukturen überhaupt offen im Sinne Ecos sein können, also Interpretationsspielräume lassen und formuliert es in [22, S. 147] folgendermaßen:

> Nun zeichnen sich künstlerische Arbeiten aber dadurch aus, dass kein Werk dem Betrachter genau das mitteilen kann, was der Künstler beabsichtigt. Voraussetzung für die echte Partizipation des Betrachters ist, dass jedes Werk Lesarten ermöglicht, die von der Intention des Entwicklers und seinen Formalisierungen abweichen. Es stellt sich die Frage, wo beim technischen Medium Computer der Freiraum für die Partizipation und die Imagination des Betrachters und die Unschärfen der Programmierung liegen.

Daraufhin definiert Trogemann die Offenheit von Software ganz allgemein als Eigenschaft, „dem Benutzer bzw. Rezipienten jeweils neue Handlungs- und Interpretationsspielräume zu eröffnen und eine gewisse Flexibilität und Durchlässigkeit für dessen Intentionen zu schaffen.“ [22, S. 146]

4.2 Das Medium

Unter einem Medium versteht man allgemein das in der Mitte Befindliche. Es dient „als vermittelndes Element zur Weitergabe und Verbreitung von Bedeutungen, Informationen und Botschaften“ [21, S. 150] zwischen einem Sender und einem Empfänger. Laut [21, S. 150] gibt es drei grundlegende Funktionen der Medien:

- Daten speichern,
- Daten übertragen,
- Daten verarbeiten.

Ein Medium als Mittel der Übertragung dient somit vorrangig der Kommunikation in Form von Rede, Schrift, Buch, Film, Fernsehen und Internet (vgl. [21, S. 150f]).

Allerdings ist dies nur ein Aspekt der Definition von Medium. Ein weiterer eröffnet sich mit Marshall McLuhans provokativer Behauptung, das „Medium ist die Botschaft“ und nicht die vom Medium übertragenen Botschaften. Am Beispiel des elektrischen Lichts erklärt er in [16, S. 22] den Zusammenhang der Medien aus seiner Sicht:

> Elektrisches Licht ist reine Information. Es ist gewissermaßen ein Medium ohne Botschaft, wenn es nicht gerade dazu verwendet wird, einen Werbetext Buchstabe um Buchstabe auszustrahlen. Diese für alle Medien charakteristische Tatsache bedeutet, daß der „Inhalt“ jedes Mediums immer ein anderes Medium ist. Der Inhalt der Schrift ist Sprache, genauso wie das geschriebene Wort Inhalt des Buchdrucks ist und der Druck wieder Inhalt des Telegrafen ist.

Medien sind laut McLuhan also ineinander verschachtelt und so zumindest auch immer ein Teil der Botschaft. Außerdem sieht er in der „Botschaft“ jedes Mediums oder jeder Technik auch „die Veränderung des Maßstabs, Tempos oder Schemas, die es der Situation des Menschen bringt.“ [16, S. 22f] Er verdeutlicht es in [16, S. 23] am Beispiel der Eisenbahn als Medium:

> Die Eisenbahn hat der menschlichen Gesellschaft nicht Bewegung, Transport oder das Rad oder die Straße gebracht, sondern das Ausmaß früherer menschlicher Funktionen vergrößert und beschleunigt und damit vollkommen neue Arten von Städten und neue Arten der Arbeit und Freizeit geschaffen. Und das traf zu,

> ob nun die Eisenbahn in einer tropischen oder nördlichen Umgebung fuhr, und ist völlig unabhängig von der Fracht oder dem Inhalt des Mediums Eisenbahn.

Es geht also nicht um die „Botschaft“ des Mediums selbst, sofern es überhaupt eine transportiert, sondern um die Veränderung, die das Medium mit sich bringt, wie im Falle der Eisenbahn die neuen Arten von Städten sowie die damit verbundenen Arbeits- und Freizeitmöglichkeiten. In Bezug auf das elektrische Licht sieht McLuhan die Verwendungsmöglichkeiten von Licht als dessen „Inhalt“ an, weil sie ohne diesen nicht sein könnten. Deshalb bestimmt das Medium auch „Ausmaß und Form des menschlichen Zusammenlebens“ [16, S. 23]. Auf den Inhalt im Speziellen kommt es allerdings nicht an, sondern auf dessen Wirkung und dessen Bedeutung, was man also durch den Einsatz von elektrischem Licht erreicht (vgl. [16, S. 23]). Mit einem Beispiel der Firma IBM[1] verdeutlicht McLuhan in [16, S. 23f] seine These:

> Als IBM entdeckten, daß ihre Tätigkeit nicht die Erzeugung von Bürobedarfoder Büromaschinen ist, sondern die Verarbeitung von Information, begannen sie, ihr Unternehmen mit klarem Blick zu leiten.

Betrachtet man den Computer als Medium, wird die von McLuhan beschriebene Verschachtelung ebenfalls deutlich. Unsere Computer vereinen heute zahlreiche klassische Medien wie Zeitungen, Radio, Fernsehen bzw. Texte, Fotos, Videos und Tonaufnahmen in einer virtuellen Welt. Dies ist allerdings nur der Anfang, denn die neuen computerbasierten Medien sind zu weit mehr fähig als ihre klassischen Vorgänger, die sie zunächst nachbilden. Die interaktiven und sozialen Möglichkeiten des Internets werden bereits heute auf den Internetplattformen der Zeitungen, Radios und Fernsehsender eingesetzt und erweitern deren Angebote. Die Botschaften und Informationen, die uns über das Internet erreichen, sind ebenfalls Medien und auch Quelltext ist ein Teil dieser Botschaften, was in Abschnitt 4.4 näher erläutert wird.

4.3 Der Code

4.3.1 Definition

Code entstammt dem lateinischen Wort „Codex“, was ursprünglich „Verzeichnis, Urkunde, Hausbuch“ bedeutete. Im 19. Jahrhundert wurde der Begriff als Fachausdruck der Fernmeldetechnik und des militärischen Nachrichtenwesens zur Ver- und Entschlüsselung von Nachrichten verwendet (vgl. [21, S. 45]).

[1] International Business Machines Corporation, US-amerikanisches IT- und Beratungsunternehmen.

Codes finden in vielen verschiedenen Bereichen Anwendung, prinzipiell dienen sie laut [19, S. 11] aber immer mindestens einem der drei folgenden Zwecke:

- Kommunikation,
- Erklärung,
- Verschleierung.

Als weiteren Zweck kann man außerdem noch die Kompression von Daten hinzufügen. Hier werden Codierungen genutzt, um Speicherplatz zu sparen, Datenraten zu verkleinern und Übertragungszeiten zu verkürzen.

Der Morsecode wandelt Buchstaben, Zahlen und Zeichen in kurze und lange Licht- oder Tonsignale um. So können sie z. B. mit Lampen oder per Funk übertragen und beim Empfänger wieder decodiert werden. Das zeigt auch den Ablauf eines Übertragungsvorgangs auf. Der Sender encodiert seine Nachricht nach bestimmten Regeln und erhält den entsprechenden Code, den er an den Empfänger sendet. Dieser nutzt nach der Übertragung das gleiche Regelwerk, um die Nachricht wieder zu decodieren. Der Empfänger wandelt den Code also zurück in eine Sprache, die er versteht.

In der Programmierung und demnach auch in der generativen sowie interaktiven Kunst codiert man die Algorithmen in Quellcode. Ein Algorithmus ist dabei eine Abfolge von eindeutig beschriebenen Befehlen, die ausführbar und in mehreren Aspekten endlich sein müssen. Diese Endlichkeiten beziehen sich auf den Text, der die Befehle beschreibt, den Speicherplatz, den sie benötigen, und die Schritte, die nötig sind, um sie auszuführen (vgl. [24], s. auch [22, S. 180ff]).

Diese formale Definition ist allerdings wenig relevant für generative Künstler. Für sie sind in [19, S. 13] praktischere Definitionen aufgeführt:

- Es gibt viele Wege, einen Algorithmus zu schreiben. Für jedes Problem gibt es also mehrere Lösungsansätze, die unterschiedliche Vor- und Nachteile mit sich bringen.
- Ein Algorithmus benötigt Annahmen bzw. Voraussetzungen. Jedes Problem und jeder Befehl bringt unterschiedliche Anforderungen mit sich, die man kennen und erfüllen muss. Diese können formaler (syntaktisch) oder inhaltlicher (semantisch) Natur sein.
- Ein Algorithmus enthält Entscheidungen. In Abhängigkeit der Eingabeparameter kann der Algorithmus verschiedene Ergebnisse liefern. Er muss also auf die möglichen Eingaben eingestellt sein und entscheiden können, wie sie weiterverarbeitet werden.
- Ein komplexer Algorithmus sollte modular aufgeteilt werden. Eine Aufteilung des Problems in viele kleine Teilprobleme macht die Problemlösung übersichtlicher und einfacher. Dies gilt auch für die Strukturierung von Programmcode. Umso modularer er aufgebaut ist, desto einfacher kann man die Teilaufgaben bearbeiten, verändern oder erweitern.

Bei allen Definitionen ist der Code immer Medium zwischen zwei Systemen und dient der Übersetzung in ein Zielsystem. In der Kommunikationswissenschaft wird der Code selbst auch als Sprache bezeichnet (vgl. [36]). Auch in Programmiersprachen dient der Quellcode der Kommunikation zwischen Mensch und Maschine. Er besitzt ebenso grammatische und syntaktische Regeln, jedoch lässt er im Vergleich zu einer menschlichen Sprache keine Interpretationen zu. Jeder Befehl muss genau definiert sein. Jeder kleinste Schreibfehler führt zu einem Fehler in der Maschine wohingegen Menschen in der Lage sind, auch fehlerhafte Texte bis zu einem gewissen Grad richtig zu interpretieren (vgl. [19, S. 15]).

Code ist das Material des generativen Künstlers. In [22, S. 42] wird deutlich, wie programmierende Künstler mit Code arbeiten und wie sich der Prozess der Softwareerstellung gestaltet:

> Die direkte Auseinandersetzung mit dem Code als Material und die Widerstände, die sich erst im Vorgang des Programmierens zeigen und überwunden werden müssen, sind hierbei genauso wichtig wie das Endprodukt. Der Prozess der Softwareerstellung ist in künstlerischen Zusammenhängen sehr viel persönlicher, die Sicht- und Herangehensweisen sind subjektiver. Es ist nicht wie in der Informatik möglich, zunächst eine Spezifikation der Software vorzulegen, die es dann erlaubt, die eigentliche Programmerstellung den Codieren zu überlassen. Programmierende Künstler begreifen den Code als Material und damit als wesentliche Komponente ihrer Projekte. Sie können sich nicht vorstellen, diese Aufgabe ohne Athentizitätsverlust für ihre Arbeiten an andere zu delegieren.

Dennoch ist es vorstellbar, dass Künstler mit Programmierern zusammenarbeiten. Je nach Intention des Künstlers ist es mit Sicherheit auch möglich, dass er ausschließlich ein Konzept bereitstellt bzw. Regie führt und, ähnlich den Konzeptkünstlern, das Codieren delegiert. Dies kann ebenso ein künstlerischer Prozess sein, in dem er in ständigem Kontakt mit dem Programmierer steht und diesen Prozess gemeinsam mit ihm erlebt. Wie ein Regisseur ist der Künstler dann der geistige Urheber des Werks.

Die ersten code-gesteuerte Maschinen wurden mit Hilfe von Lochkarten betrieben. Durch den Einsatz von Codes begann so die Ablösung der Software von der Hardware (vgl. [22, S. 74]). Die Tragweite dieses Wandels wird in [22, S. 22] deutlich:

> Während die Hardware an die physikalischen Naturgesetze gebunden ist, unterliegt die Software nur noch den Grenzen der eigenen Vorstellungskraft.

Eine Alternative zu Quellcode bieten visuelle oder grafische Programmiersprachen wie Max oder vvvv (Abb. 4.3). Sie verwenden an Stelle von Text

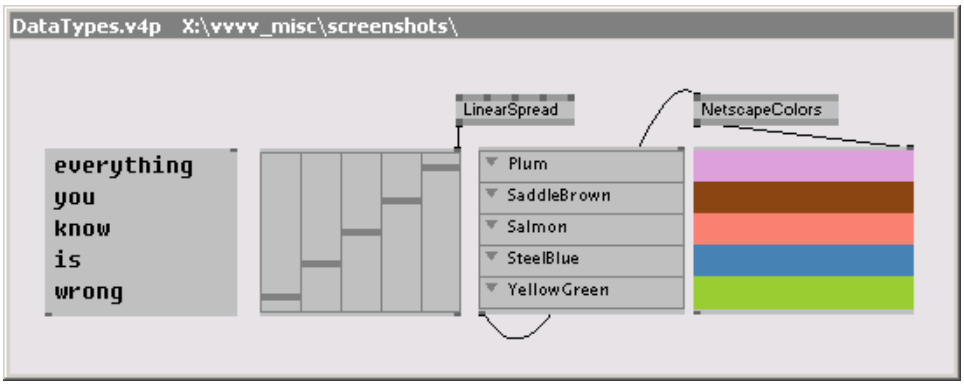


Abbildung 4.3: Vier verschiedene Datentypen aus vvvv. Bildquelle: [67].

grafische Elemente, genannt „Nodes“, die Teilprobleme wie z. B. einfache Rechenoperationen oder Transformationen lösen. Sie besitzen Eingänge, die Daten empfangen, und Ausgänge, die das Teilergebnis an andere Nodes weitergeben. Diese Weitergabe erfolgt durch Verknüpfungen, die durch den Programmierer erstellt werden. Die verknüpften Nodes bilden letztlich eine Befehlskette und somit auch einen Algorithmus.

Bildlich gesprochen ist Code mittlerweile nicht mehr nur in unseren Computern und auf unseren Bildschirmen. Wir spüren ihn überall um uns herum, in Geräten und Installationen, in Objekten aus Holz, Metall und Plastik, die mit Hilfe von computergesteuerten Maschinen produziert wurden und in Daten oder Texten, die ausgedruckt und somit von der virtuellen in die physische Welt übertragen werden (vgl. [19, S. 23]).

4.3.2 Nach Flusser

Vilém Flusser bedient sich in seiner Definition von Code dem Begriff des Symbols. Er sieht Symbole als Phänomene, die andere Phänomene ersetzen. Das scheint vorerst nicht viel auszusagen, soll aber bedeuten, dass dieser Ersatz eine Form der Kommunikation ist. Ein Erlebnis selbst wird also durch die Kommunikation durch seinen Sinn und seine Bedeutung ersetzt. Ein System aus Symbolen ergibt laut Flusser dann einen Code. Menschen bedienen sich solcher Codes, um miteinander zu kommunizieren, da sie die eigentliche Bedeutung der Symbole nicht mehr kennen. Flusser sieht im Menschen ein „verfremdetes Tier“, das Symbole schafft und sie in Codes ordnet, um der Welt eine Bedeutung zu geben. Beispielsweise sind von Menschenaffen vor zwei Millionen Jahren geschaffene Kreise aus Bärenknochen und Steinen für uns heute nicht mehr erklärbar. Wir kennen den Schlüssel zu diesen Codes nicht und wissen demnach auch nicht, was diese Symbole, die Steine und Knochen, bedeuten, aber wir erkennen die Künstlichkeit darin, was uns auf menschliche Gegenwart schließen lässt (vgl. [7, S. 23]).

Außerdem listet Flusser in [8, S. 77] verschiedene Arten von Symbolen auf, die zu Codes geordnet werden können:

- punktartig (Mosaik),
- geradlinig (Alphabet),
- Wellenlinien (Arabeske),
- flächenartig (Zeichnungen und Gemälde),
- oberflächlich (Teppich),
- körperlich (Skulptur und Architektur),
- vierdimensional (Tanz und Gesten),
- räumlich (Drahtmodell),
- zeitlich (Musik) und
- noch komplexere Dimensionsverbindungen (Film, Theater, Lichtreklame und Verkehrszeichen).

Die Komplexität unserer „kodifizierten Welt“ ist allerdings viel zu groß, um sie gänzlich zu durchblicken und Flusser warnt davor, die Codes unserer Welt der Kulturen zu katalogisieren und sie so entziffern zu wollen. Des Weiteren führt er drei charakteristische Codes auf, die sich im Laufe der Zeit entwickelt und die im Folgenden näher erläutert werden (vgl. [8, S. 77f]).

Das Vor-Alphabet

Unser heutiges Alphabet ist aramäisch und man begann in der ersten Hälfte des zweiten Jahrtausends vor Christus damit, unsere Buchstaben zu schreiben. Diese Buchstaben scheinen ursprünglich allerdings keine Töne gewesen zu sein, sondern Gegenstände. Beispielsweise bedeutet „Alpha“ auf aramäisch „Ochse“, „Beta“ bedeutet „Haus“ und „Gama“ bedeutet „Kamel“. Es sind ursprünglich also ebenso aneinandergereihte Bilder, ähnlich den ägyptischen Hieroglyphen, gewesen. Die Entwicklung der einzelnen Buchstaben und Alphabete erfolgt laut Flusser in mehreren Stufen (vgl. [8, S. 86]):

1. Piktogramme,
2. Ideogramme,
3. Hieroglyphen und
4. Buchstaben.

Die einfachste Form ist das Piktogramm, welches ein vereinbartes Abbild eines Gegenstandes bedeutet. Ebenso ist ein Ideogramm ein Abbild eines Gegenstandes, jedoch bedeutet es nicht den Gegenstand selbst, sondern eine „allgemeine Situation“ und kann so auch als Metapher eingesetzt werden. Ein Ideogramm von einem Herzen kann beispielsweise Liebe bedeuten. Bei Hieroglyphen wird die Bedeutung weiter abstrahiert indem es ein Wort in einer gegebenen Sprache darstellt. Der abgebildete Gegenstand muss allerdings nicht in Bezug zu der Wortbedeutung stehen. Flusser führt an dieser

Stelle das Beispiel mit dem Bild eines Zehs auf, der ebenso „zäh“ bedeuten kann. Letztlich handelt es sich bei Buchstaben des Vor-Alphabets nicht mehr um Abbildungen von Gegenständen, sondern um die ersten Töne der Worte, die sie beschreiben, z. B. ist „A“ der erste Ton des Wortes „Alpha“. Flusser sieht in diesen Vorstufen jedoch keine Entwicklung, sondern eher „eine andere Form von Verfremdung“ und jede dieser Formen kann „zu einem anderen Höhepunkt, zu einer anderen Existenzform, einer anders kodifizierten, aber gleich ‚hochstehenden‘ (gleichermaßen verfremdeten) Welt beziehungsweise Kultur führen.“ [8, S. 87] Dennoch haben sie alle eine Gemeinsamkeit: sie sind „lineare“ Codes, d. h., ihre Symbole sind in Zeilen bzw. Reihen geordnet. Sie werden nacheinander gelesen und werden so zu einer Erzählung, während ein normales Bild mit den gleichen Piktogrammen eine Szene darstellt. Der Unterschied liegt darin, dass der Text eine Geschichte erzählt, also einen Prozess bedeutet (vgl. [8, S. 86ff]).

Das Alphabet

Die Buchstaben des Alphabets werden je nach Sprache anders gesprochen. Flusser führt weiterhin aus, dass es sich dabei nicht um tatsächliche Laute handelt, sondern um „die Konvention eines Lauts“, welche „Folge einer orthographischen Konvention“ ist. [8, S. 89f] Es gibt noch weitere Konventionen, die sich auf die gesprochene und die schriftliche Sprache beziehen, wichtig ist jedoch, dass die Bedeutung der Buchstaben generell von Konventionen abhängig sind und sie das Alphabet zu einem sehr abstrakten Code machen (vgl. [8, S. 89f]).

Das Alphabet wurde geschaffen, um Sprachverwirrungen innerhalb der Elite aufzuheben. Diese alphabetisierte Elite bestand aus Händlern und Kaufleuten, die Lager- und Ladelisten, Abrechnungen sowie Kalkulationen mit Hilfe dieses neuen Codes austauschten (vgl. [8, S. 91f]). Flusser behauptet, dieser alphabetische Code „ist ein für das Rechnen und Zählen, Wiegen und Messen vereinbarter Code.“ [8, S. 92] Flusser bezeichnet das Alphabet daraufhin in [8, S. 93] als Code der Elite, wohingegen die Bilder der Code des Volkes sind:

> Sie ist vom Kampf zwischen Alphabet (dem Code der Elite) und Bildercodes (den Codes des „Volkes“) gekennzeichnet, also dem Kampf zwischen einem rechnenden („historischen“) und einem imaginierenden („magischen“) Bewußtsein. Kalkulus gegen Ritual, Begriff gegen Vorstellung, Buchstabe gegen Bild – das ist das Thema dieses Zeitabschnitts, von der prä-sokratischen Philosophie über die Scholastik bis zum Humanismus. Das ist der „Kern der Geschichte“.

Das Nach-Alphabet

Flusser sieht nun einen weiteren Wandel des Codes, wie es auch schon vom Vor-Alphabet zum Alphabet, von Bildern zu alphabetischen Programmen geschehen ist. Für ihn besteht das folgende Nach-Alphabet aus Technobildern, in welche wir das Alphabet übersetzen. Technobilder stellen nach Flusser die „Welt der Fotografien, der Filme, der Fernsehschirme und roten Ampeln“ [8, S. 100] dar und er beschreibt sie in [8, S. 103] wie folgt:

> Der Fotograf steht „hinter“ dem Schriftsteller, welcher „hinter“ dem Zeichner steht, welcher „hinter“ der Welt steht. Um zu zeichnen, muß man von der Welt Abstand nehmen. Um zu beschreiben, muß man von Bildern Abstand nehmen. Um zu fotografieren, muß man über eine Beschreibung verfügen. Der Code der Technobilder ist ein nachalphabetischer Code und hätte ohne das Alphabet nicht erfunden werden können.

Mit der Zeit lernen wir den Umgang mit diesem neuen Code und begreifen seine neuen Dimension. Wir lösen uns von der Linearität der Texte und suchen nach Wegen, mehr als nur Geschichten zu erzählen (vgl. [8, S. 104f]).

Computer und die damit verbundenen Ausdrucksformen, also auch generative und interaktive Kunst, sind Teil dieser Technobilder, dessen Anwendung wir mehr und mehr zu verstehen beginnen. Somit ist auch der Quellcode ein Teil des Nach-Alphabets, wenn auch nicht in der gleichen Bedeutung, wie Flusser den Begriff Code in seinen Ausführungen verwendet. Dennoch kann man den Quellcode ebenso als ein eigenständiges Alphabet sehen, welches geschaffen wurde, um die Kommunikation zwischen Menschen und Maschinen zu ermöglichen, so wie das Alphabet die Kommunikation zwischen den Händlern und Kaufleuten ermöglicht hat.

4.3.3 Code ist Poesie

Abgeleitet von dem englischen Statement „CODE IS POETRY“ in der Fußzeile der offiziellen WordPress-Webseite[2], soll dieser Abschnitt sich mit der Ästhetik von Quellcode und Programmierung auseinandersetzen. In [22, S. 43] schreibt Georg Trogemann dazu Folgendes:

> Programmierung ist Begegnung mit sich selbst und mit entäußerten Gedanken und Vorstellungen. Ähnlich dem literarischen Schreiben werden Gedanken in Reihenfolge gebracht und schriftlich festgehalten. Den niedergeschriebenen, entäußerten Gedanken begegnen wir im Falle eines normalen Textes beim Lesen wieder. In der Programmierung begegnen wir unseren Vorstellungen in der Ausführung in Form von Prozessen oder errechneten Ergebnissen wieder.

[2] http://www.wordpress.org

Der Prozess, den man während des Programmierens durchläuft, ähnelt also dem des literarischen Schreibens. Er ist kreativ. Man muss die richtigen Worte bzw. Befehle finden, welche die anfängliche Idee zum Algorithmus und schließlich zu geschriebenem Quelltext machen. Dieser wird dann zwar nicht gelesen, aber vom Computer ausgeführt und vom Rezipienten interpretiert.

Matt Ward schreibt in [71] über „The Poetics Of Coding“. Er wurde dazu ebenfalls vom oben genannten Wordpress-Statement inspiriert und vergleicht in erster Linie die Struktur von Gedichten mit der Struktur von Webtechnologien wie HTML, CSS und Javascript. Er formuliert es u. a. folgendermaßen:

> Code has purpose and meaning. It requires structure. It should be lightweight and elegant, not bogged down with lines and lines of garbage. Writing great code isn't something that just happens. It takes discipline and work! It's an art unto itself.
>
> [...] Perhaps code really is a form of poetry, and the coder a new kind of poet.

Im Gegensatz dazu kommentiert Adit Gupta unter dem Artikel, dass die wahre Schönheit von Quellcode nicht in der äußeren Struktur, sondern vielmehr in den Ideen, Algorithmen und Strategien liegt, die in ihm enthalten sind:

> A truly beautiful code will amaze you at how brilliantly some complex functionality was achieved or how beautifully an algorithm was implemented. The structure and clarity is like make-up, which does not essentially represents the true beauty of a code.

Ebenso wird in [38] deutlich, dass die Ästhetik von Quellcode wie auch bei der Poesie in der Ausführung liegt, nicht ausschließlich in der geschriebenen Form:

> Code itself is clearly not poetry as such, but retains some of its rhythm and metrical form. Code is intricately crafted, and expressed in multitudinous and idiosyncratic ways. Like poetry, the aesthetic value of code lies in its execution, not simply its written form.

Eindeutiger bestätigt Matt Pearson in [17, S. xxxviii] die Behauptung, dass Code Poesie ist, wenn wir bereit sind, damit zu experimentieren und kommerzielle Programmierung außer Acht lassen:

> My conjecture is that code can be poetry, and code can be fun. But we may have to sacrifice some of the rigidity, good design, and best practices of professional, commercial programming to enable this.

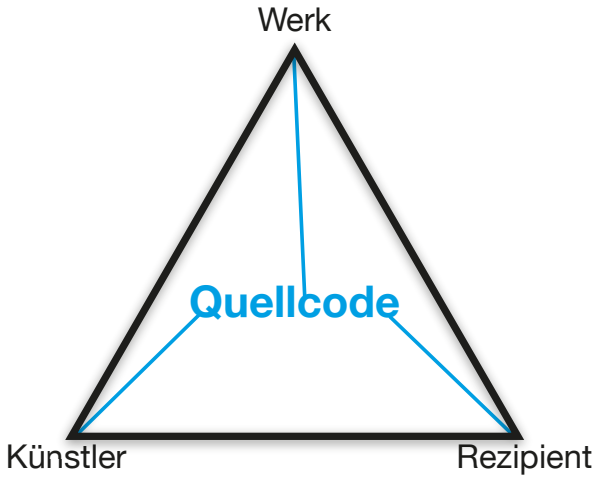


Abbildung 4.4: Der Quellcode als Medium zwischen Künstler, Werk und Betrachter.

4.4 Schlussfolgerung

Im letzten Abschnitt dieses Kapitels soll nun erörtert werden, inwieweit der Quellcode als Medium funktioniert. Georg Trogemann sieht dazu in [22, S. 43f] das Kunstwerk als einen „kollaborativen Vorgang“, der erst durch programmierbare Medien zustande kommt:

> Zwar stellt sich diese Frage schon lange vor der Verwendung avancierter Technologien in der Kunst, aber erst durch programmierbare Medien kann sich auch die logische und ästhetische Struktur des Kunstwerks verändern, und zwar überprüfbar und nicht nur in der Interpretation des Betrachters. Das Kunstwerk soll zu einem kollaborativen Vorgang werden, an dem sich Künstler, Rezipienten und nicht zuletzt die Maschine beteiligen.

Diese von Trogemann beschriebenen interaktiven Kunstwerke sind also erst mit dem Einsatz von Quellcode als Medium möglich. Der Code positioniert sich in genau in der Mitte des klassischen Dreiecks und vermittelt zwischen allen drei Punkten (Abb. 4.4).

Die Bedeutung des Codes für die Kunst kann jedoch in Zukunft noch wesentlich größer sein und muss nicht allein auf die Interaktion mit dem Rezipienten beschränkt sein. Trogemann macht in [22, S. 44f] auf viele Arten deutlich, welches Potential in der künstlerischen Programmierung steckt, was im Folgenden auszugsweise dargestellt werden soll:

> Wer die Sprache des Codes zu lesen in der Lage ist, dem steht

> das Medium Computer in seiner ganzen Multiperspektivität offen, die weit über den bloßen Werkgebrauch hinausreicht und sich auch nicht in den ewigen Variationen bekannter Wahrnehmungsformen erschöpft. Erst innerhalb des Horizontes der Codes können alle künstlerischen Strategien und Herangehensweisen konsequent durchgespielt werden.
>
> [...] Wir können gezielt Störungen einbauen oder das Medium demontieren, um die Eigenstrukturen sichtbar zu machen oder den konstruktiven Charakter des Mediums in den Vordergrund zu stellen. Wir können den Code selbst zum Material erklären und diesen in seiner Prozesshaftigkeit oder seiner Zeichenhaftigkeit, oder beides zusammen untersuchen. Mit dem Code steht uns auch das ganze Repertoire der Mittel zur Verfügung, um den Mediengebrauch in sich zu verändern und erweitern. Wir können mit der Maschinenhaftigkeit des Mediums spielen und diese hervorheben oder umgekehrt die Mechanik dissimilieren.
>
> [...] Erst wenn Künstler die mächtige Form der Programmierung beherrschen, können sie beginnen, ihre eigenen Werkzeuge zu entwickeln.

Der Quellcode ist somit quasi der Pinselstrich des programmierenden Künstlers und das macht ihn zum Werkzeug, ohne welches ein generatives bzw. interaktives Kunstwerk nicht erschaffen werden kann. Diese Behauptung setzt, wie in Kapitel 1 erwähnt, voraus, dass generative und interaktive Werke ausschließlich mit der Hilfe von Computern erzeugt werden.

In Bezug auf die Rezeption gibt es zwei Aspekte. Zum einen kann ein interaktives Werk nicht ohne Quellcode präsentiert werden, da nur seine Ausführung auf einem Rechner das eigentliche Kunstwerk darstellt. Zum anderen ist bei generativen Werken der Quellcode zur Präsentation nicht zwangsläufig notwendig, da auch die Möglichkeit besteht, dass sie durch einen endlichen Algorithmus erstellt wurden und sie somit z. B. auch in Printform unveränderlich präsentiert werden können. In diesem Fall steht der Quellcode nicht mehr als Medium zwischen Werk und Rezipient.

Die Frage, die noch offen bleibt, ist die Frage nach dem Unterschied zwischen Quellcode und Werk. Doch auch hier kann man den Quellcode als Mittel und Werkzeug vom Werk als Ergebnis und letztendlicher Rezeption trennen.

Kapitel 5

Werkanalyse – Ars Rata

Im folgenden Kapitel soll eine Werkanalyse des Projekts Ars Rata durchgeführt werden. Dazu wird zunächst das Konzept des Projekts vorgestellt und die technischen Aspekte, vor allem in Bezug auf die generative Erzeugung und Animation, beschrieben. Anschließend werden die generativen Elemente in Bezug zu den in Kapitel 3 beschriebenen Methoden gesetzt und letztlich die Frage geklärt, inwieweit Ars Rata generativ ist.

5.1 Konzept

5.1.1 Idee

Ars Rata ist ein VFX-Kurzfilm, welcher Kunst im Ausstellungsraum thematisiert. Dazu wird eine dreidimensionale kinetische Skulptur gerendert und in einen realen Raum eingebettet. Der Ausstellungsraum des Offenen Kulturhauses in Linz weist hierbei auch einen Bezug zur Kunst auf, da dort regelmäßig Ausstellungen zeitgenössischer Kunst stattfinden.

Die kinetische Skulptur wird generativ erzeugt und nimmt vorwiegend abstrakte Formen an. Dabei dienen bereits vorhandene Formen von Skulpturen aus der bildenden Kunst als Inspiration (Abb. 5.1–5.3).

5.1.2 Ablauf und Intention

Der Kurzfilm beinhaltet keine narrativen Elemente, aber dennoch gibt es eine Dramaturgie innerhalb der Formen, der Animationen und des Schnitts (Abb. 5.4). Zusätzlich nimmt die Skulptur unterschiedliche Aggregatzustände an. Sie tritt z. B. auch teilweise in flüssiger Form auf und wird mit Hilfe von Kraftfeldern animiert.

Zu Beginn ist eine Kugel aus metallähnlichem Material am Boden des Ausstellungsraums zu sehen, gefilmt mit einer weitwinkligen Handkamera, die das Geschehen zu dokumentieren versucht. Die Kugel beginnt zu schweben und nähert sich der Mitte des Raumes. Dabei verflüssigt sie sich und

Abbildung 5.1: Olafur Eliasson: Model room, 2003. Bildquelle: [27].

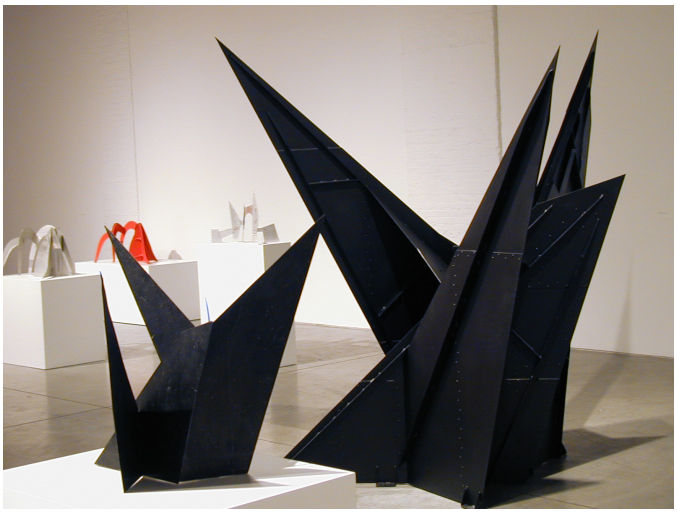


Abbildung 5.2: Alexander Calder: Gwenfritz (zwei Maquetten), Blech, Schrauben und Farbe, 1964 und 1968. Bildquelle: [28].

Abbildung 5.3: Anish Kapoor: Non-Object (Spire), Edelstahl, 2007. Bildquelle: [50].

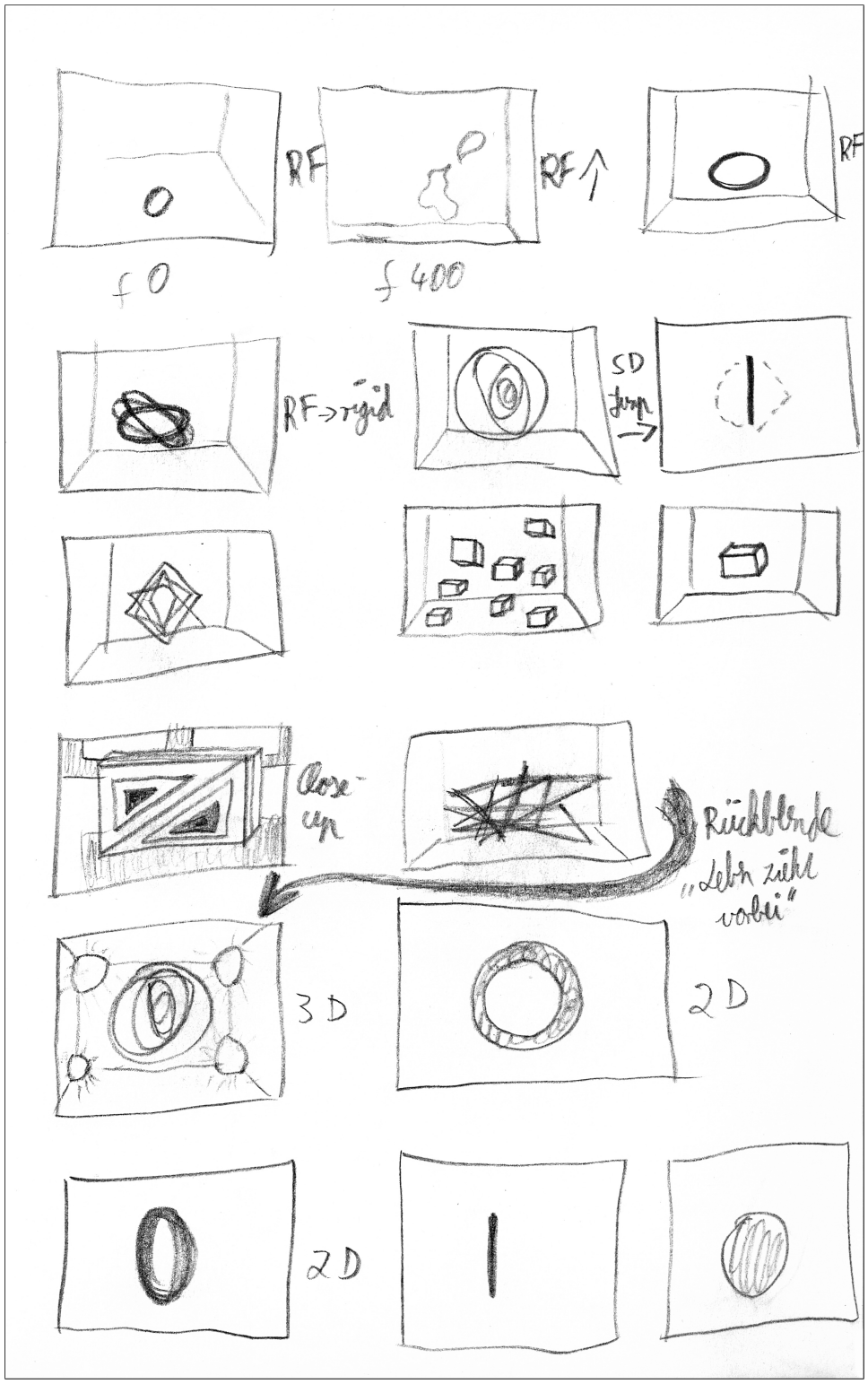


Abbildung 5.4: Storyboard Ars Rata.

bildet tentakelähnliche Arme. In der nächsten Stufe teilt sich der flüssige Ball und bildet eine zweite Kugel über sich, mit der er im Anschluss kollidiert. Diese Kollision zu Beginn wirkt durch die anschließende ringförmige Ausbreitung der Flüssigkeit wie ein Urknall und definiert das Ende des Intros.

Im Anschluss bilden sich acht rigide Segmente aus dem flüssigen Ring heraus und symbolisieren so die Geburt der kinetischen Skulptur. Die Segmente werden geometrisch im Raum transformiert bis sie in der Lebensmitte der Skulptur einen Ruhepol in Form eines schwebenden Würfels finden. Dieser hat zwar im Gegensatz zur Kugel am Anfang schon Ecken und Kanten, aber er besitzt dennoch eine simple Geometrie, die Ruhe ausstrahlt. Sie repräsentiert die sprichwörtliche „Ruhe vor dem Sturm“, der im Anschluss mit spitzen Dreiecken eingeleitet wird.

Das Leben der Skulptur nähert sich ihrem Ende und ihr letztes Aufbäumen gegen diese Gefahr soll durch die aggressive Form der langen Zacken verkörpert werden. Für eine kurze Zeit scheint diese Abwehrhaltung Wirkung zu zeigen während sich die Skulptur langsam im Raum dreht, ihre Spitzen im Raum verteilt und dem Betrachter dabei gefährlich nah kommt. Das Ende ist jedoch nicht aufzuhalten und beginnt mit der Auflösung des Lebensraumes der Skulptur. Der rechteckige Raum scheint von einer großen Kugel förmlich gefressen zu werden, die beginnend an den Ecken den Raum unausweichlich verschlingt. Gleichzeitig symbolisiert das eintretende Weiß das Licht am Ende des Tunnels, welches aus Nahtoderfahrungen oft geschildert wird.

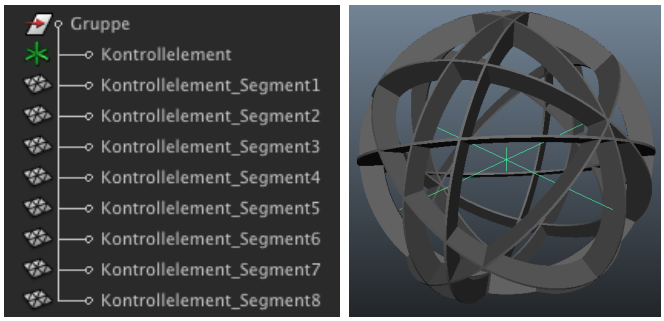
Während sich dieser dreidimensionale Raum also zunehmend in einen weißen Unraum verwandelt, durchläuft auch die Skulptur eine Transformation. Ihre Formen sind Wiederholungen aus Formen ihres vorangegangenen Lebens, die sich auf metaphorische Weise erneut vor ihrem inneren Auge abspielen. Ihr metallisches Material löst sich zunehmend auf und wird von einem Schwarz überschattet, das keine Struktur oder Räumlichkeit aufweist und so den Gegensatz zum einfallenden Weiß bildet.

Was bleibt ist ein zweidimensionales Bild aus einer schwarzen Skulptur und einem weißen Raum, der jegliche Tiefe verloren hat. Die Skulptur wird zu einer Fläche, die einen Ring symbolisch für die Unendlichkeit bildet.

5.2 Technik und Programmierung

Die Geometrie der kinetischen Skulptur ergibt sich aus acht Polygon Primitiven vom Typ „polyTorus“. Diese bilden die acht Segmente und können über Parameter in Radius, Drehung und Unterteilung sogar so weit modifiziert werden, dass sie neben den runden Formen auch zu den drei- und viereckigen führen.

Gesteuert und animiert werden diese acht Objekte über ein zentrales



(a) Hierarchie **(b)** Skulptur

Abbildung 5.5: Screenshots der Hierarchie und der Skulptur in Autodesk Maya.

Kontrollelement, welches über Expressions[1] mit ihnen verbunden ist. Das Kontrollelement ist ein Locator, der zusätzlich zu den Transformationsparametern (Translation, Rotation, Skalierung) über ein Pythonskript[2] mit weiteren Parametern versehen wurde, um die oben genannten Toruseigenschaften der acht Segmente zu kontrollieren.

Die Rotation des Kontrollelements führt zu einer unregelmäßigen Rotation der einzelnen acht Segmente auf Basis von statischen Faktoren (Seeds), die zuvor vom Pythonskript zufällig erzeugt wurden und zwischen den Segmenten variieren. Je nach Faktor drehen sich die Segmente also unterschiedlich weit. Für eine globale Rotation der gesamten Skulptur wurde deshalb eine Gruppe erzeugt, die das Kontrollelement und die dazugehörigen Segmente zusammenfasst (Abb. 5.5).

5.2.1 Animationsparameter

Die folgenden Parameter können in der Animation von Ars Rata vom Künstler über das Kontrollelement genutzt werden:

Translate X, Y, Z: Verschiebt alle Segmente gleichmäßig im Raum.

Rotate X, Y, Z: Dreht jedes Segment mit einem zufällig generierten Faktor um die eigenen drei Achsen.

Scale X, Y, Z: Skaliert jedes Segment gleichmäßig in allen drei Dimensionen.

Radius: Legt den Radius aller Segmente um das Kontrollelement fest.

Radius Offset: Erweitert oder reduziert den Radius der Segmente in gestaffelter Form, relativ zum Kontrollelement.

[1] s. auch Abschnitt 5.2.3.
[2] s. auch Abschnitt 5.2.2.

Section Radius: Legt den lokalen Radius um die Achse fest, die durch das Segment verläuft.

Twist: Dreht jedes Segment entlang der Achse, die durch das Segment verläuft.

Subdivision Axis: Bestimmt die vertikale Unterteilung/Auflösung jedes Segments.

Subdivision Height: Bestimmt die horizontale Unterteilung/Auflösung jedes Segments.

Offset X, Y, Z: Verschiebt die Segmente gestaffelt im Raum, relativ zum Kontrollelement.

Random Offset X, Y, Z: Verschiebt jedes Segment mit einem zufällig generierten Faktor im Raum.

Jedes Segment besitzt für die zufälligen Transformationen drei statische Zufallszahlen, die innerhalb der Animationsparameter als Faktoren eingesetzt und im Abschnitt 5.3.5 näher erläutert werden.

5.2.2 Pythonskript

Python[3] ist eine quelloffene und plattformunabhängige Programmiersprache der Python Software Foundation. Die Quelloffenheit und Plattformunabhängigkeit sind zwei wichtige Kriterien dafür, dass es zunehmend als Skriptsprache in größere Softwarepakete eingebunden wird. So z. B. auch in der für Ars Rata verwendeten 3D-Software Autodesk Maya. Die kinetische Skulptur des Projekts wird innerhalb der Software über ein Pythonskript erzeugt, welches im Folgenden in Verbindung mit Codeabschnitten näher erläutert wird.

```
1 import maya.cmds as cmds
2 import random
```

Um in Autodesk Maya interne Funktionen wie z. B. das Erstellen eines Polygons über Python aufrufen zu können, muss die Bibliothek „maya.cmds“ importiert werden. Für die Erzeugung der Zufallszahlen später im Skript wird außerdem die Bibliothek „random“ geladen.

```
1 def createRotator(name = None, rings = 0):
```

Die erste von zwei Methoden im Skript namens „createRotator“ erstellt das Kontrollelement, welches innerhalb des Skripts als „rotator“ bezeichnet wird. Es können zwei optionale Parameter an die Methode übergeben werden. Zum einen der Name des Kontrollelements, falls mehrere Kontrollelemente erstellt werden, und zum anderen die Anzahl der Segmente, die im Skript als „rings“ bezeichnet werden, da sie in ihrer Grundform als PolyTorus ringförmig sind. Wird der Parameter Name nicht übergeben, wird er nicht gesetzt und so ein Standardname von Autodesk Maya vergeben. Falls die Segmentanzahl nicht

[3] http://www.python.org/

gesetzt ist, wird ausschließlich das Kontrollelement erstellt und vorerst keine Segmente daran gebunden. Diese Bindung kann über den Aufruf der zweiten Methode des Skripts auch später erfolgen.

```
1     locator = cmds.spaceLocator(n = name)[0]
```

Zu Beginn wird das Kontrollelement generiert. Dazu wird ein Locator erstellt und, falls ein Name übergeben wurde, benannt. Der erstellte Locator wird in der gleichnamigen Variablen „locator“ gespeichert, um im nachfolgenden Code weiterhin auf dieses Element zugreifen zu können.

Ein Locator in Autodesk Maya ist ein Objekt, welches beliebig dreidimensional transformiert werden kann. Allerdings dient diese Objekt ausschließlich als Hilfsobjekt, um beispielsweise Positionen im Raum zu markieren oder auch zusätzliche Parameter zu speichern, die keine Transformationsparameter sind. Diese zusätzlichen Parameter, genannt Attribute, werden im Anschluss mit der Funktion „addAttr“ erstellt.

```
1     cmds.addAttr(longName = "radius", attributeType = "double", \
2                  keyable = True, minValue = 0.001, defaultValue = 16)
3     cmds.addAttr(longName = "radiusOffset", attributeType = "double", \
4                  keyable = True, defaultValue = 0)
5     cmds.addAttr(longName = "sectionRadius", attributeType = "double", \
6                  keyable = True, minValue = 0.001, defaultValue = 1)
7     cmds.addAttr(longName = "twist", attributeType = "double", \
8                  keyable = True, defaultValue = 45)
9     cmds.addAttr(longName = "subdivisionsAxis", \
10                 attributeType = "short", keyable = True, \
11                 minValue = 3,defaultValue = 64)
12    cmds.addAttr(longName = "subdivisionsHeight", \
13                 attributeType = "short", keyable = True, \
14                 minValue = 3, defaultValue = 4)
15
16    cmds.addAttr(longName = "offsetX", attributeType = "double", \
17                 keyable = True, defaultValue = 0)
18    cmds.addAttr(longName = "offsetY", attributeType = "double", \
19                 keyable = True, defaultValue = 0)
20    cmds.addAttr(longName = "offsetZ", attributeType = "double", \
21                 keyable = True, defaultValue = 0)
22    cmds.addAttr(longName = "randomOffsetX", attributeType = "double", \
23                 keyable = True, defaultValue = 0)
24    cmds.addAttr(longName = "randomOffsetY", attributeType = "double", \
25                 keyable = True, defaultValue = 0)
26    cmds.addAttr(longName = "randomOffsetZ", attributeType = "double", \
27                 keyable = True, defaultValue = 0)
```

Die aufgerufenen Funktionen beziehen sich automatisch auf den erstellten Locator, weil dieser nach seiner Erstellung automatisch von Autodesk Maya selektiert wird und die Funktion „addAttr“ auf dem selektierten Objekt ein neues Attribut erstellt. Somit muss der Locator bzw. dessen Variable innerhalb der Funktion nicht extra angegeben werden.

Die Attribute werden mit einem Namen und einem Datentypen erstellt, der bestimmt, welche Werte das Attribut speichert. In diesem Fall sind die Attribute vom Datentyp „double“, also Fließkommazahlen, und „short“, also ganze Zahlen. Des Weiteren werden die Attribute mit Standardwerten und zum Teil mit Minimal- und Maximalwerten erzeugt, sodass sie diesen angegebenen Bereich nicht über- oder unterschreiten. Der Parameter „keyable“ der Funktion „addAttr“ wird bei allen Attributen auf „True“ gesetzt, sodass die erstellten Attribute in der Channel Box von Autodesk Maya angezeigt werden und für den Benutzer einfacher modifizierbar sind.

```
1     locatorshape = cmds.listRelatives(shapes = True)[0]
2     locatorshape = cmds.rename(locatorshape, locator + "Shape")
3     cmds.expression(s = locatorshape + ".localScaleX = " \
4                     + locator + ".radius;\n" \
5                     + locatorshape + ".localScaleY = " \
6                     + locator + ".radius;\n" \
7                     + locatorshape + ".localScaleZ = " \
8                     + locator + ".radius;\n")
```

Anschließend wird noch eine Expression dem Kontrollelement hinzugefügt, welche die Größe des Locators in Beziehung zum Radiusparameter setzt. Vergrößert der Benutzer also den Radius wird auch der Locator vergrößert und der Benutzer erhält ein direktes visuelles Feedback auf seine Aktion. Technisch muss dazu auf „LocalScale“ des Locators zugegriffen werden. Dieser Wert befindet sich in der Shape des Locators, weshalb diese zuvor in der Variablen „locatorshape“ über die Funktion „listRelatives“ gespeichert und anschließend in der Expression wieder referenziert wird.

```
1     createRing(locator = locator, rings = rings)
```

Abschließend wird die zweite Methode aufgerufen, welche die Segmente erstellt. Dazu wird ihr das erstellte Kontrollelement und die Anzahl der zu erstellenden Segmente übergeben. Die zweite Methode wird im Folgenden näher beschrieben.

```
1 def createRing(locator = cmds.ls(selection = True), rings = 1):
```

Die Methode „createRing“ zur Erstellung der Segmente besitzt ebenfalls zwei optionale Parameter. Zum einen wird ein Kontrollelement benötigt, an das eine beliebige Anzahl Segmente gebunden wird. Wird kein Konotrollelement in Form eines Locators übergeben, wird das Objekt verwendet, das gerade in Autodesk Maya selektiert ist. Dieses selektierte Objekt muss allerdings ein Locator sein. Der zweite Parameter bestimmt die Anzahl der zu erstellenden Segmente. Ist er nicht gesetzt, wird genau ein Segment an das Kontrollelement gebunden.

```
1     if rings < 1 or not len(locator): return
2     if type(locator) is list: locator = locator[0]
3     if not cmds.objectType(cmds.pickWalk(locator, direction = "down")) \
```

```
4        == "locator": return
```

Zu Beginn der Methode erfolgt eine Überprüfung der übergebenen bzw. gesetzten Parameter. Die Segmentlänge in den Variablen „rings“ darf beispielsweise nicht kleiner als eins sein und es muss ein Locator übergeben bzw. vor der Ausführung des Skripts selektiert worden sein. Anschließend erfolgt die Überprüfung, ob die Selektion auch ein Locator und kein anderes 3D-Objekt ist.

```
1    for i in range(0,rings):
```

Anschließend folgt eine Schleife, die über die gegebene Anzahl an Segmenten entsprechend PolyTori anlegt.

```
1        iname = str(locator) + "_ring1"
2        itorus = cmds.polyTorus(n = iname, sr = 1, sx = 32, sy = 4)
3
4        iname = itorus[0]
5        itorus = itorus[1]
6
7        ai = int(iname.rsplit("_ring", 1)[1]) - 1
```

Zunächst wird in der Variablen „iname“ der Name des Segments festgelegt, welcher sich aus dem Namen des Kontrollelements und der Endung „_ring1“ zusammensetzt. In der folgenden Zeile wird das Segment als PolyTorus mit diesem Namen erstellt. Sollte der Name bereits existieren, z. B. bei einem weiteren Schleifendurchlauf, wird die Ziffer am Ende des Namens automatisch von Autodesk Maya inkrementiert. Das Segment des zweiten Durchlaufs endet somit mit „_ring2“.

Danach wird für weitere Referenzierungen der richtige Name, sofern automatisch inkrementiert wurde, in der Variablen „iname“ gespeichert. Der Zugriff erfolgt über „itorus“, welches nach der Erstellung eine Liste aus zwei Elementen von der Funktion „polyTorus“ erhält. Das erste Element der Liste ist der Name des Objekts und das zweite Element der Knoten in Autodesk Maya. „itorus“ wird anschließend ebenfalls für weitere Referenzierungen mit diesem Knoten überschrieben.

Die Variable „ai“ ergibt sich anschließend aus dem Vorgänger der Segmentnummer und wird später für die Staffelung der Segmente in den Parametern „offsetX“, „offsetY“, „offsetZ“ und „radiusOffset“ verwendet. Weitere Ausführungen zur Staffelung der Segmente sind in Abschnitt 5.2.3 unter den genannten Parametern zu finden.

```
1        cmds.addAttr(longName = "seedX", attributeType = "double", \
2                     keyable = True, defaultValue = random.random())
3        cmds.addAttr(longName = "seedY", attributeType = "double", \
4                     keyable = True, defaultValue = random.random())
5        cmds.addAttr(longName = "seedZ", attributeType = "double", \
6                     keyable = True, defaultValue = random.random())
```

Nach der Erstellung des PolyTorus' ist dieser automatisch selektiert. Daraufhin werden mit der Funktion „addAttr“ und der Bibliothek „random“ drei Zufallszahlen für jede Achse im dreidimensionalen Raum generiert, sodass nach allen Schleifendurchläufen jedes Segment drei eigene Zufallszahlen besitzt. Diese werden dann als Faktoren in den Translations- und Rotationsparametern eingesetzt, um eine zufällige Verteilung und Drehung im Raum zu implementieren.

```
1         cmds.expression(s = [...])
```

Am Ende jedes Schleifendurchlaufs werden die Expressions für die Transformationsparameter und die PolyTorus-Attribute der Segmente gesetzt. Sie verwenden die vorher definierten Variablen, um die in Abschnitt 5.2.3 beschriebenen Funktionen zu realisieren. Der vollständige Quelltext ist in Anhang B zu finden.

```
1     cmds.select(locator)
```

Der abschließende Befehl der Methode außerhalb der beschriebenen Schleife ist die Selektion des mit Segmenten versehenen Kontrollelements.

5.2.3 Expressions

Expressions sind Ausdrücke einer Skriptsprache, die Attribute eines Objekts modifizieren. Mit Hilfe dieser Ausdrücke können Parameter über mathematische Operationen oder Variablen (z. B. die aktuelle Bildnummer in der Zeitleiste) bzw. Konstanten (z. B. Pi) verändert werden. Außerdem können Parameter auch über Expressions miteinander verbunden werden (vgl. [26]). Beispielsweise liegen sie im Projekt Ars Rata auf allen Transformationsparametern jedes einzelnen Segments und verbinden diese mit dem Kontrollelement. Es folgt ein Beispiel des achten Segments:

```
1 rotator1_ring8.translateX = rotator1.translateX + (rotator1.
      randomOffsetX * (rotator1_ring8.seedX - 0.5) * 2) + (rotator1.
      offsetX * 7);
2 rotator1_ring8.translateY = rotator1.translateY + (rotator1.
      randomOffsetY * (rotator1_ring8.seedY - 0.5) * 2) + (rotator1.
      offsetY * 7);
3 rotator1_ring8.translateZ = rotator1.translateZ + (rotator1.
      randomOffsetZ * (rotator1_ring8.seedZ - 0.5) * 2) + (rotator1.
      offsetZ * 7);
4 rotator1_ring8.rotateX = rotator1.rotateX * rotator1_ring8.seedX;
5 rotator1_ring8.rotateY = rotator1.rotateY * rotator1_ring8.seedY;
6 rotator1_ring8.rotateZ = rotator1.rotateZ * rotator1_ring8.seedZ;
7 rotator1_ring8.scaleX = rotator1.scaleX;
8 rotator1_ring8.scaleY = rotator1.scaleY;
9 rotator1_ring8.scaleZ = rotator1.scaleZ;
```

Die drei Translate-Parameter ergeben sich aus der Position des Kontrollelements „rotator1“. Außerdem wird der Faktor „randomOffset“ mit der jeweiligen Zufallszahl „seed“ des Segments multipliziert und für die zufällige Verteilung im Raum zum Translate-Parameter hinzugefügt. Abschließend wird für die gestaffelte Verteilung im Raum „offset“ addiert. Um die Staffelung zu implementieren, wird „offset“ mit einem Faktor multipliziert, der sich aus dem Vorgänger der Nummer des Segments ergibt. So besitzt z. B. Segment acht den Faktor sieben und das erste Segment den Faktor null. Beim ersten Segment wird somit nicht gestaffelt durch „offset“ verschoben.

Die Rotation bezieht sich ebenfalls auf das Kontrollelement indem es dessen Rotation verwendet. Diese Rotation wird allerdings noch mit der Zufallszahl des Segments multipliziert, um eine Varianz der Rotation unter den Segmenten zu erreichen.

Die Skalierung hingegen wird direkt vom Kontrollelement übernommen, sodass diese für jedes Segment gleichermaßen gilt, ohne den Einsatz von zufälligen Faktoren.

Zusätzlich werden auch Eigenschaften der erstellen PolyTori über Expressions gesteuert, die ebenfalls Parameter des Kontrollelements verwenden. Der folgende Code bezieht sich erneut auf das achte Segment bzw. auf die Attribute seines PolyTorus:

```
1 polyTorus8.radius = rotator1.radius + rotator1.radiusOffset * 7;
2 polyTorus8.sectionRadius = rotator1.sectionRadius;
3 polyTorus8.twist = rotator1.twist;
4 polyTorus8.subdivisionsAxis = rotator1.subdivisionsAxis;
5 polyTorus8.subdivisionsHeight = rotator1.subdivisionsHeight;
```

Der Radius des Segments ergibt sich aus dem Radiusparameter des Kontrollelements und dem „radiusOffset“, welcher mit einem Faktor multipliziert wird. Dieser Faktor ist wie bei „offset“ in den Transformationsparametern von der Segmentnummer abhängig und sorgt ebenfalls für eine Staffelung.

Die übrigen Parameter werden direkt an die Attribute des Kontrollelements gebunden und gelten für alle Segmente gleichermaßen. Ihre Funktionen wurden in Abschnitt 5.2.1 bereits erläutert.

5.3 Generative Elemente

Ars Rata weist den Großteil der in Kapitel 3 vorgestellten generativen Techniken auf. In den folgenden Abschnitten werden diese generativen Aspekte des Projekts aufgeführt und in den Kontext des Konzepts gesetzt.

5.3.1 Wiederholung

Die kinetische Skulptur besteht aus technischer Sicht zu jeder Zeit, mit Ausnahme des flüssigen Zustands, aus acht Segmenten. Die Wiederholung besteht somit primär aus einer physischen Wiederholung der Segmente. Sie

Abbildung 5.6: Acht Segmente mit zufälligen Offsets.

weisen zwar die gleichen Eigenschaften auf und sind alle gleichermaßen an dasselbe Kontrollelement gebunden, können aber dennoch zum gleichen Zeitpunkt durch definierte und zufällige Offsets[4] unterschiedliche Formen annehmen (Abb. 5.6).

Eine zeitliche Wiederholung entsteht außerdem durch die Dramaturgie in der Animation. Am Ende, während der Raum sich auflöst, wiederholen sich die Schlüsselformen der Skulptur in umgekehrter Reihenfolge. Diese Wiederholung wird bewusst eingesetzt, um das vorherige Leben der Skulptur noch einmal in Kurzform zu visualisieren während seine Umgebung durch das Auflösen und mit dem sprichwörtlichen strahlenden Licht am Ende des Tunnels in einem Nichts zu enden scheint.

5.3.2 Transformation

Die Bewegung der kinetischen Skulptur im Raum besteht in Ars Rata ausschließlich aus einer geometrischen Transformation. Die Segmente der Skulptur werden dreidimensional bewegt, gedreht und skaliert. Die Rotation ist in diesem Fall die wichtigste geometrische Transformation. Durch die Drehung der Segmente um die eigene Achse und die Achse des Kontrollelements im Zentrum wird der Großteil der Dramaturgie umgesetzt. Ein prägnanter Einsatz der Skalierung erfolgt beim Wandel der Skulptur von einem Würfel zu den aggressiven dreieckigen Spitzen, die sich schnell im Raum ausbreiten (Abb. 5.7). Translation kommt hingegen nur sehr dezent zum Einsatz, beispielsweise in Form eines leichten Schwebens der Skulptur über die gesamte

[4] engl., Versatz.

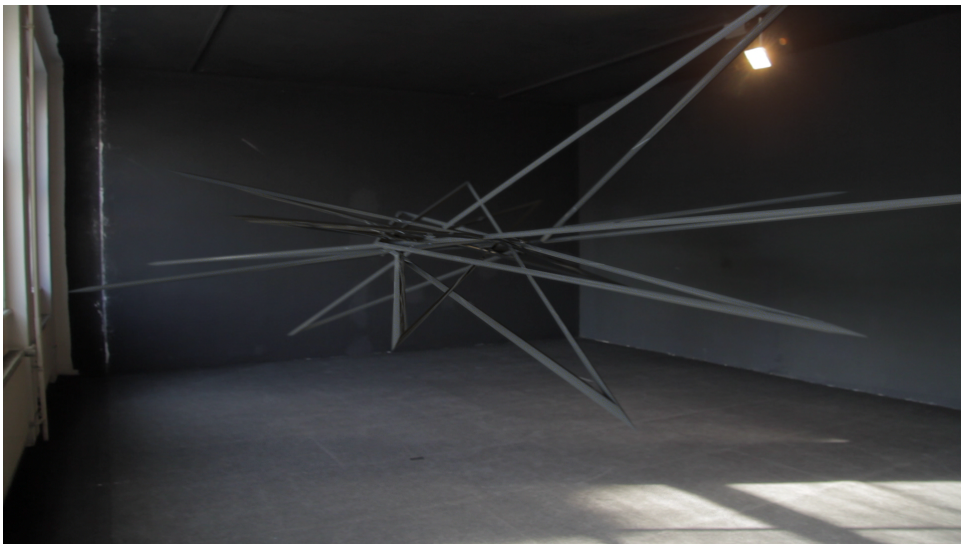


Abbildung 5.7: Unproportional skalierte Dreiecke, die mit Hilfe zufälliger Faktoren rotiert wurden.

Animation hinweg. Die Translation der Segmente entsteht größtenteils durch die Rotation um den zentralen Kontrollpunkt und durch Offset-Animationen, die bestimmen, wie weit sich die Segmente vom Zentrum entfernen.

Eine Bildverarbeitung in Form einer numerischen Transformation findet im generativen Sinn nicht statt. Ebenso wird keine Transkodierung verwendet, die beispielsweise externe Daten als Parameter für die Animation einspeist oder den Ton visualisiert. Die Animation besteht rein aus manuell gesetzten Keyframes auf dem Kontrollelement, welches die Bewegung der Segmente bestimmt.

5.3.3 Simulation

Der Aspekt der Simulation ist ebenfalls im analysierten Werk vertreten. Die kinetische Skulptur wechselt in mehreren Momente den Aggregatzustand zwischen fest und flüssig. Die liquiden Formen wurden mit Hilfe von Flüssigkeitssimulationen erzeugt. Dazu wurden zunächst Partikel innerhalb einer Ausgangsform erzeugt und anschließend mit virtuellen Kräften, die meist physikalischen Kräften nachempfunden sind, in Bewegung versetzt.

Die Partikel besitzen zueinander und gegenüber von anderen Objekten in der Szene ein spezielles Verhalten. So wird z. B. die Oberflächenspannung von Flüssigkeiten berücksichtigt, indem nahe beieinander liegende Partikel sich gegenseitig anziehen. Sie bilden dadurch Gruppen und nehmen einzelne Partikel auf solange keine stärkeren Kräfte auf sie wirken und sie wieder zerreißen. Außerdem kollidieren sie miteinander und gegenüber anderen

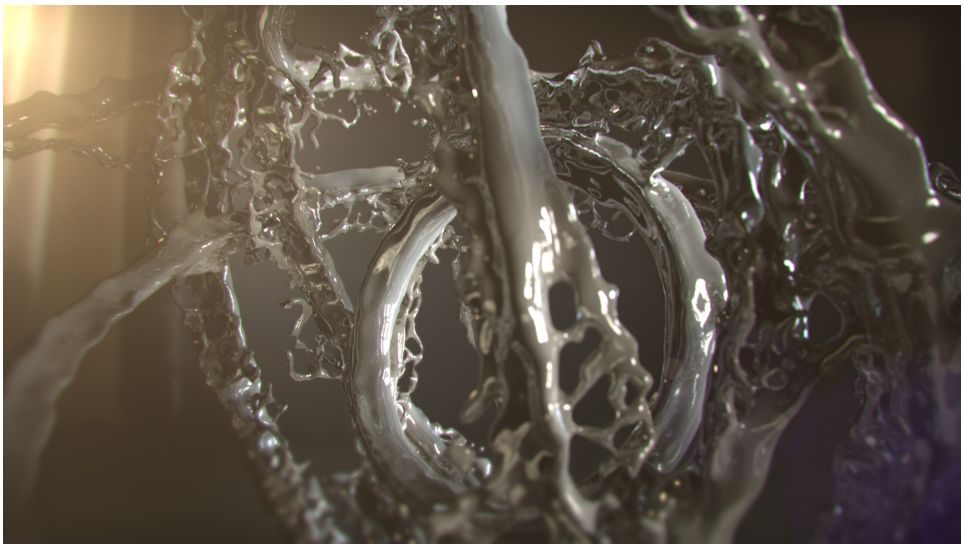


Abbildung 5.8: Verflüssigte kinetische Skulptur.

Objekten, die evtl. auch andere Materialeigenschaften besitzen können. Beispielsweise prallen Flüssigkeitspartikel stärker von einem festen Körper ab als von anderen Flüssigkeitspartikeln, in die sie eher eintauchen oder mit denen sie sich vermischen.

Nach der Simulation der Partikel mit virtuellen Kräften und Kollisionen werden die Partikel anhand ihrer Position im Raum miteinander verrechnet und so eine zusammenhängende Form (Mesh[5]) generiert. Hierbei kann über Parameter bei z. B. Größe und Glättung noch Einfluss auf die finale Form genommen werden.

In Ars Rata dient der liquide Aggregatzustand vor allem dazu, die Lebendigkeit der Skulptur zum Ausdruck zu bringen und sie organischer wirken zu lassen. Am Ende wird eine nahe Einstellung in Zeitlupe gezeigt, worin die Skulptur sich ebenfalls verflüssigt (Abb. 5.8). Das erweckt den Eindruck, dass der eigentliche flüssige Zustand der Skulptur erst durch die Hochgeschwindigkeitsaufnahme sichtbar wird und sie bei schnellen Transformationen auseinander reißt. Vergleichbar ist diese Nahaufnahme mit Hochgeschwindigkeitsaufnahmen von z. B. einem Boxkampf, bei dem erst durch die Zeitlupe sämtliche Verformungen bei einem Treffer von der Kamera bzw. dem Auge erfasst werden. Im Anschluss wird wieder in die Totale mit normaler Geschwindigkeit geschnitten, sodass die Skulptur wieder als feste Form wahrgenommen wird.

[5] engl., Netz, Geflecht.

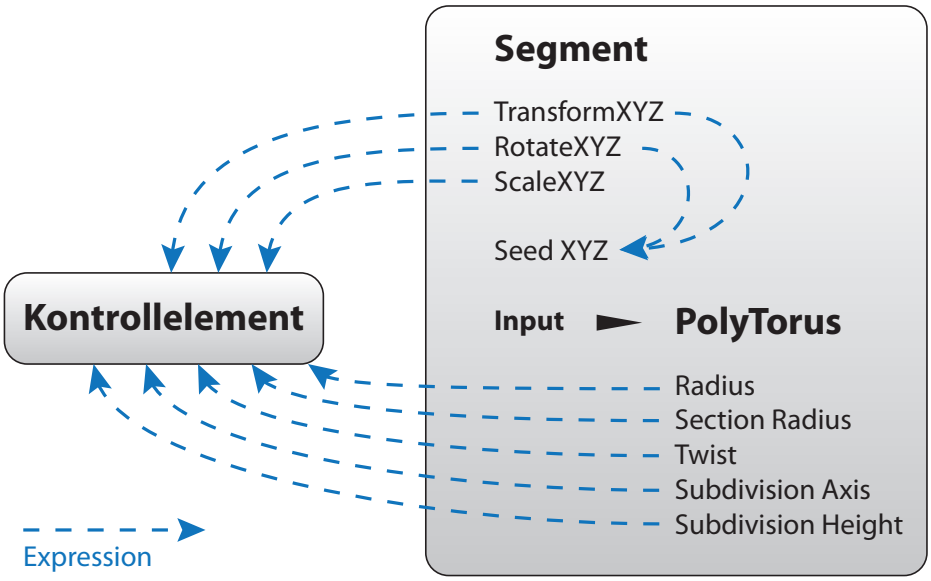


Abbildung 5.9: Parameter der Segmente werden durch Expressions bestimmt, die größtenteils auf Parameter des Kontrollelements verweisen.

5.3.4 Parametrisierung

Die gesamte Animation von Ars Rata basiert in erster Linie auf Parametrisierung. Das Konzept sieht ein zentrales Kontrollelement vor, welches die acht Segmente über verschiedenste Parameter bewegt, dreht und skaliert. Realisiert wird diese Parametrisierung über Expressions, die auf allen Transformationsparametern der Segmente liegen (Abb. 5.9). Die Segmente werden somit nur indirekt bewegt. Ändert sich ein Parameter des Kontrollelements, werden dann über die Expressions die Transformationen für die Segmente berechnet und angewandt.

Beispielsweise ergibt sich die Transformation eines Segments zunächst aus der Transformation des Kontrollelements. Außerdem werden auf diese Transformation noch das lineare Offset und das zufällige Offset addiert. Das zufällige Offset für ein Segment ergibt sich dabei aus den Offsetwerten im Kontrollelement sowie den Seed-Faktoren, welche für jedes einzelne Segment zufällig generiert wurden. Ebenso findet diese Verrechnung mit den Seed-Faktoren bei der Rotation der Segmente statt.

5.3.5 Zufall

Obwohl durch die Verwendung von Keyframes die vollständige Kontrolle der Animation beim Gestalter liegt, spielt der Zufall eine entscheidende Rolle. Durch den statischen Einsatz von Zufallszahlen ist es möglich, die Segmente

voneinander zu trennen ohne eine Regelmäßigkeit oder ein geordnetes System darzustellen. Die Zufallszahlen wurden bei der Erstellung des Kontrollelements und der acht Segmente für jedes einzelne Segment für drei Dimensionen generiert. Jedes Segment besitzt somit für jede Achse im dreidimensionalen Raum eine eigene Zufallszahl. Sie werden in den unter Abschnitt 5.3.4 beschriebenen Expressions als Faktoren verwendet, um die Transformationen eines Segments von den anderen Segmenten zu unterscheiden.

Das hat zur Folge, dass beispielsweise eine Rotation um die Y-Achse des Kontrollelements zu einer Rotation aller Segmente führt, die allerdings mit ihren eigenen Y-Zufallszahlen multipliziert werden und somit alle eine unterschiedliche Rotation aufweisen. Trotzdem stehen diese Rotationen jedoch immer noch in Verbindung mit dem Kontrollelement.

Kapitel 6

Zusammenfassung

> While the industrial revolution produced tools to augment the body, such as the steam engine and the automobile, the information revolution is producing tools to extend the intellect. [19, S. 17]

Generative und interaktive Kunstwerke sind ein Teil dieser Erweiterung unseres Intellekts und sie werden sich in Zukunft noch stärker in der zeitgenössischen Kunst etablieren. Die Bewegung STEAM[1], geführt von John Maeda, zeigt außerdem, dass auch die Industrie in Zukunft stärker mit Kunst und Design in Verbindung stehen wird. Der Name dieser Bewegung basiert auf dem Kurznamen STEM für Science, Technology, Engineering und Math als klassische Begriffe für Innovation. STEAM fügt einen weiteren Aspekt hinzu: „Art/Design“ und setzt sich zum Ziel, die Integration von Kunst und Design in der Wirtschaft zu verstärken.

Diese Master Thesis definierte zu Beginn die Begriffe generative sowie interaktive Kunst und setzte sie in historischen Kontext. Außerdem wurde zum Abschluss der Einführung die Definition und Relevanz des Zufalls für generative und interaktive Werke erläutert. Des Weiteren wurden Techniken und Erscheinungsformen aufgezeigt, die belegen, wie vielfältig der Ausdruck generativer sowie interaktiver Kunstwerke sein kann. Gleichzeitig bildeten diese Ausdrucksformen die Grundlage für die spätere Werkanalyse. In dieser fand außerdem eine Erläuterung des Konzepts und eine Dokumentation des Projekts statt. Abschließend wurde evaluiert, inwieweit das Projekt Ars Rata generativ ist.

Laut den unter Kapitel 2 aufgeführten Definitionen ist es essentiell, einen Teil der Verantwortung an ein externes System abzugeben. Das hat zur Folge, dass das Ergebnis unvorhersehbar ist und, wie bei der Konzeptkunst, vom Künstler nicht beeinflusst werden kann. In Ars Rata wurde diese Verantwortung nur temporär abgegeben. Zwar ist beispielsweise die Verteilung

[1] http://stemtosteam.org/

der Elemente vorerst dem Zufall überlassen, jedoch wurde diese zufällige Verteilung so lange verändert, bis sie in das vordefinierte Konzept passte. Dieses Vorgehen entspricht in etwa der Methodik von Dadaist Hans Arp[2], der zunächst ebenfalls seine Farbpapiere zufällig auf einen Karton fallen lies, diese dann jedoch im Anschluss noch nach eigenen künstlerischen Kriterien verschob. So wurde die Verantwortung zu Beginn an den Zufall als externes System abgegeben, anschließend wurde sie ihm durch den manuellen Eingriff aber auch wieder entzogen.

Das Projekt weist somit zwar generative Methoden auf, die Verantwortung bleibt aber dennoch durch die durchgehende Kontrolle beim Künstler. Das System, welches generativ arbeitet, dient in diesem Fall zwar als Werkzeug, jedoch nicht als selbstständiger Mitgestalter des Werks.

Ein entscheidender Faktor hierfür sind die verwendeten Keyframes. Sie werden allein vom Künstler gesetzt und bilden die Grundlage jeder Animation im Werk. Um den Prozess generativer zu gestalten, müssten diese Keyframes von einem externen System generiert werden. Das hätte zur Folge, dass jede Bewegung der kinetischen Skulptur und der Ausgang der Animation ungewiss wäre.

Allerdings entspricht diese Vorgehensweise nicht dem grundlegenden Konzept von Ars Rata, welches versucht, eine Dramaturgie zu verfolgen und eine ästhetische Erfahrung zu vermitteln, die aus einer Kombination von realen Kameraaufnahmen, einer computergenerierten Skulptur, Videoschnitten und Audio besteht. Der generative Aspekt ist somit nur ein Teil des Gesamtwerks, welches komplexer ist als nur die Animation der kinetischen Skulptur.

Die Master Thesis befasste sich des Weiteren mit dem Quellcode als Medium zwischen Künstler, Werk und Rezipient. Dadurch soll auch eine Betrachtung des Themas aus medientheoretischer Sicht sichergestellt sein. Die Bedeutung des Codes steigt vor allem mit der Präsenz des Internets in unserem täglichen Leben. Denn umso mehr das Internet die klassischen Medien wie Fernsehen, Radio oder Printmedien verdrängt bzw. in sich vereint, desto mehr wird auch der Quellcode als Medium, das Informationen sammelt, verarbeitet und präsentiert, an Bedeutung gewinnen.

In Bezug auf Ars Rata war der Quellcode des in Abschnitt 5.2.2 beschriebenen Pythonskripts entscheidende Grundlage für die Umsetzung des Projekts. Durch das Skript wurde der Grundaufbau der Skulptur mit ihren acht Segmenten und einem Kontrollelement für die Animation generiert. Dieser Aufbau macht den primären Stil des Kurzfilms aus. Zwar wäre eine Umsetzung dieser Animation dennoch auch ohne Programmierung möglich gewesen, allerdings mit einem wesentlich höheren Zeit- und Arbeitsaufwand. Dieser Punkt führt zurück zu dem einleitenden Zitat von Joshua Davis in Kapitel 1. Viele generative Werke können in der Theorie problemlos auch ohne den Einsatz von Computern produziert werden, jedoch macht in der

[2] s. auch Abschnitt 2.4.

Praxis dann der tatsächliche Produktionsaufwand eine Realisierung nahezu unmöglich. Deshalb brauchen wir den Code und die Maschinen, die ihn ausführen.

Ars Rata ist laut den vorstehenden Erläuterungen also kein rein generatives Kunstwerk, aber es weißt Merkmale generativer Kunst auf. Das ist ein weiterer Beweis für die Allgegenwärtigkeit generativer Elemente, die über viele Bereiche der Kunst hinweg vertreten sind. Es ist also auch weniger eine Frage, ob Kunstwerke generativ sind, sondern nur in welchem Ausmaß. Generative und interaktive Elemente werden demnach in Zukunft noch stärker in unser Leben rücken, sowohl in Kunst und Kultur als auch in Industrie und Technik.

Anhang A

Inhalt der CD

Format: CD-ROM, Single Layer, ISO9660-Format

A.1 Master Thesis

Pfad: /

Masterarbeit.pdf	Diese Master Thesis „Generative und interaktive Computerkunst mit dem Quellcode als Medium“ in digitaler Form

A.2 Abbildungen

Pfad: /Abbildungen/

. . . . . .	In der Master Thesis eingebundene Abbildungen

A.3 Online-Quellen

Pfad: /Online-Quellen/

. . . . . .	Archivierte Online-Quellen

A.4 Projektdateien

Pfad: /Projektdateien/

Pythonskript.py	Das in Abschnitt 5.2.2 erläuterte Pythonskript
Weblink OSX.webloc . .	OSX-Link zum Kurzfilm Ars Rata
Weblink Windows.url . .	Windows-Link zum Kurzfilm Ars Rata

Anhang B

Quelltext des Pythonskripts

```
1  import maya.cmds as cmds
2  import random
3
4
5  def createRotator(name = None, rings = 0):
6      locator = cmds.spaceLocator(n = name)[0]
7
8      cmds.addAttr(longName = "radius", attributeType = "double", \
9                   keyable = True, minValue = 0.001, defaultValue = 16)
10     cmds.addAttr(longName = "radiusOffset", attributeType = "double", \
11                  keyable = True, defaultValue = 0)
12     cmds.addAttr(longName = "sectionRadius", attributeType = "double", \
13                  keyable = True, minValue = 0.001, defaultValue = 1)
14     cmds.addAttr(longName = "twist", attributeType = "double", \
15                  keyable = True, defaultValue = 45)
16     cmds.addAttr(longName = "subdivisionsAxis", \
17                  attributeType = "short", keyable = True, \
18                  minValue = 3,defaultValue = 64)
19     cmds.addAttr(longName = "subdivisionsHeight", \
20                  attributeType = "short", keyable = True, \
21                  minValue = 3, defaultValue = 4)
22
23     cmds.addAttr(longName = "offsetX", attributeType = "double", \
24                  keyable = True, defaultValue = 0)
25     cmds.addAttr(longName = "offsetY", attributeType = "double", \
26                  keyable = True, defaultValue = 0)
27     cmds.addAttr(longName = "offsetZ", attributeType = "double", \
28                  keyable = True, defaultValue = 0)
29     cmds.addAttr(longName = "randomOffsetX", attributeType = "double", \
30                  keyable = True, defaultValue = 0)
31     cmds.addAttr(longName = "randomOffsetY", attributeType = "double", \
32                  keyable = True, defaultValue = 0)
33     cmds.addAttr(longName = "randomOffsetZ", attributeType = "double", \
34                  keyable = True, defaultValue = 0)
35
36     locatorshape = cmds.listRelatives(shapes = True)[0]
37     locatorshape = cmds.rename(locatorshape, locator + "Shape")
```

```
38      cmds.expression(s = locatorshape + ".localScaleX = " \
39                      + locator + ".radius;\n" \
40                      + locatorshape + ".localScaleY = " \
41                      + locator + ".radius;\n" \
42                      + locatorshape + ".localScaleZ = " \
43                      + locator + ".radius;\n")
44
45      createRing(locator = locator, rings = rings)
46
47
48  def createRing(locator = cmds.ls(selection = True), rings = 1):
49      if rings < 1 or not len(locator): return
50      if type(locator) is list: locator = locator[0]
51      if not cmds.objectType(cmds.pickWalk(locator, direction = "down")) \
52         == "locator": return
53
54      for i in range(0,rings):
55          iname = str(locator) + "_ring1"
56          itorus = cmds.polyTorus(n = iname, sr = 1, sx = 32, sy = 4)
57
58          iname = itorus[0]
59          itorus = itorus[1]
60
61          ai = int(iname.rsplit("_ring", 1)[1]) - 1
62
63          cmds.addAttr(longName = "seedX", attributeType = "double", \
64                       keyable = True, defaultValue = random.random())
65          cmds.addAttr(longName = "seedY", attributeType = "double", \
66                       keyable = True, defaultValue = random.random())
67          cmds.addAttr(longName = "seedZ", attributeType = "double", \
68                       keyable = True, defaultValue = random.random())
69
70          cmds.expression(s = itorus + ".radius = " \
71                          + locator + ".radius + " \
72                          + locator + ".radiusOffset * " +str(ai)+ ";\n" \
73                          + itorus + ".sectionRadius = " \
74                          + locator + ".sectionRadius;\n" \
75                          + itorus + ".twist = " + locator + ".twist;\n" \
76                          + itorus + ".subdivisionsAxis = " \
77                          + locator + ".subdivisionsAxis;\n" \
78                          + itorus + ".subdivisionsHeight = " \
79                          + locator + ".subdivisionsHeight;")
80
81          cmds.expression(s = iname + ".translateX = " \
82                          + locator + ".translateX + (" \
83                          + locator + ".randomOffsetX * (" \
84                          + iname + ".seedX - 0.5) * 2) + (" \
85                          + locator + ".offsetX * " + str(ai) + ");\n" \
86                          + iname + ".translateY = " \
87                          + locator + ".translateY + (" \
88                          + locator + ".randomOffsetY * (" \
89                          + iname + ".seedY - 0.5) * 2) + (" \
90                          + locator + ".offsetY * " + str(ai) + ");\n" \
91                          + iname + ".translateZ = " \
```

```
92                          + locator + ".translateZ + (" \
93                          + locator + ".randomOffsetZ * (" \
94                          + iname + ".seedZ - 0.5) * 2) + (" \
95                          + locator + ".offsetZ * " + str(ai) + ");\n" \
96                          + iname + ".rotateX = " \
97                          + locator + ".rotateX * " \
98                          + iname + ".seedX;\n" \
99                          + iname + ".rotateY = " \
100                         + locator + ".rotateY * " \
101                         + iname + ".seedY;\n" \
102                         + iname + ".rotateZ = " \
103                         + locator + ".rotateZ * " \
104                         + iname + ".seedZ;\n" \
105                         + iname + ".scaleX = " \
106                         + locator + ".scaleX;\n" \
107                         + iname + ".scaleY = " \
108                         + locator + ".scaleY;\n" \
109                         + iname + ".scaleZ = " \
110                         + locator + ".scaleZ;")
111
112     cmds.select(locator)
```

Quellenverzeichnis

Literatur

[1] A. Alberro und B. Stimson. *Conceptual Art: A Critical Anthology*. Cambridge: MIT Press, 2000.

[2] Michael Balter. „From a Modern Human's Brow—or Doodling?“ In: *Science* 295 (Jan. 2002).

[3] Max Bense. *Aestetica: Einführung in die neue Aesthetik*. Baden-Baden: Agis-Verlag, 1982.

[4] H. Bohnacker u. a. *Generative Gestaltung: Entwerfen. Programmieren. Visualisieren*. Mainz: Verlag Hermann Schmidt, 2009.

[5] Umberto Eco und Günter Memmert. *Das offene Kunstwerk*. Frankfurt am Main: Suhrkamp, 1977.

[6] D. Elger und U. Grosenick. *Dadaismus*. Köln: Taschen, 2004.

[7] V. Flusser und S. Bollmann. *Medienkultur*. Frankfurt am Main: Fischer Taschenbuch Verlag, 1997.

[8] V. Flusser, S. Bollmann und E. Flusser. *Kommunikologie*. Frankfurt am Main: Fischer Taschenbuch Verlag, 1998.

[9] Rudolf Frieling und Dieter Daniels. *Medien Kunst Interaktion, die 80er und 90er Jahre in Deutschland*. Hrsg. von Goethe-Institut München und ZKM Karlsruhe. Berlin: Springer, 2000. URL: http://www.medienkunstnetz.de/quellentext/65/ (besucht am 01.06.2012).

[10] Florian Josef Gruber. „If Code == Imagination: Theorie, Modelle und Entwurfsmuster generativer Designprozesse“. Diss. Wien: Technische Universität Wien, Fakultät für Informatik, 2012.

[11] Christopher S. Henshilwood, Francesco d'Errico und Ian Watts. „Engraved ochres from the Middle Stone Age levels at Blombos Cave, South Africa“. In: *Journal of Human Evolution* 57 (2009), S. 27–47. URL: http://tracsymbols.eu/files/henshilwood-published-papers-for-download/Henshilwood%20et%20al%202009%20Engraved%20ochres%20from%20the%20Middle%20Stone%20Age%20levels%20at%20Blombos%20Cave%20South%20Africa%20JHE.pdf.

[12] Christopher S. Henshilwood u. a. „Emergence of Modern Human Behavior: Middle Stone Age Engravings from South Africa“. In: *Science* 295 (Feb. 2002). URL: http://tracsymbols.eu/files/henshilwood-published-papers-for-download/Henshilwood%20et%20al%202002%20BBC%20engraved%20ochre%20Science.pdf.

[13] W. Kemp. *Zeitgenössische Kunst und ihre Betrachter*. Köln: Oktagon, 1996.

[14] Ruth Leavitt. *Artist and Computer*. New York: Harmony Books, 1976.

[15] J. Linschinger und Gmundner Symposion. *Konkrete Kunst und Aleatorik: Gmundner Symposium*. Piesport: Ottenhausen-Verlag, 1995.

[16] Marshall McLuhan. *Die magischen Kanäle Understanding Media*. Dreden: Verlag der Kunst, 1995.

[17] M. Pearson. *Generative Art*. Greenwich: Manning Publications, 2011.

[18] Anna Lena Phillips. „The Algorists“. In: *American Scientist* 99.2 (2011).

[19] C. Reas und C. McWilliams. *Form+Code in Design, Art, and Architecture*. New York: Princeton Architectural Press, 2010.

[20] H. Richter. *Dada-Kunst und Antikunst: Der Beitrag zur Kunst des 20. Jahrhunderts*. Köln: M. DuMont-Schauberg, 1964.

[21] A. Roesler. *Grundbegriffe der Medientheorie*. Paderborn: Fink, 2005.

[22] G. Trogemann und J. Viehoff. *CodeArt: Eine Elementare Einführung in Die Programmierung Als Künstlerische Praktik*. Berlin: Springer, 2004.

Online-Quellen

[23] URL: http://www.pbs.org/safarchive/3_ask/archive/qna/3284_cohen.html (besucht am 26.05.2012).

[24] *Algorithmus*. URL: http://de.wikipedia.org/wiki/Algorithmus (besucht am 06.10.2012).

[25] Mark K. Anderson. *'Aaron': Art From the Machine*. 2001. URL: http://www.wired.com/culture/lifestyle/news/2001/05/43685?currentPage=all (besucht am 26.05.2012).

[26] *Animation expressions*. URL: http://download.autodesk.com/global/docs/maya2013/en_us/files/Animation_expressions_Animation_expressions.htm (besucht am 06.10.2012).

[27] A_O_G. *103g - Olafur Eliasson*. 2003. URL: http://www.flickr.com/photos/8619805@N05/3746552387/ (besucht am 06.10.2012).

[28] A_O_G. *529b - Alexander Calder*. 2006. URL: http://www.flickr.com/photos/8619805@N05/3749335765/ (besucht am 06.10.2012).

[29] ART+COM. *The Invisible Shape of Things Past*. 1995-2007. URL: http://www.artcom.de/projekte/projekt/detail/the-invisible-shape-of-things-past/ (besucht am 18.08.2012).

[30] ART+COM. *Zerseher*. 1992. URL: http://www.artcom.de/projekte/projekt/detail/zerseher/ (besucht am 01.06.2012).

[31] *ASCII art*. URL: http://en.wikipedia.org/wiki/ASCII_art (besucht am 06.10.2012).

[32] Jeremy Birn. *Color Bleeding*. 2002. URL: http://www.3drender.com/glossary/colorbleeding.htm (besucht am 12.07.2012).

[33] Christoph Braun u. a. *Beautycheck - Ursachen und Folgen von Attraktivität*. 2001. URL: http://www.beautycheck.de/cmsms/index.php/der-ganze-bericht (besucht am 31.08.2012).

[34] *Brute-Force-Methode*. URL: http://de.wikipedia.org/wiki/Brute_force (besucht am 06.10.2012).

[35] *Buckminster Fuller*. URL: http://en.wikipedia.org/wiki/R._Buckminster_Fuller (besucht am 06.10.2012).

[36] *Code*. URL: http://de.wikipedia.org/wiki/Code (besucht am 06.10.2012).

[37] *Condensation Cube*. URL: http://www.macba.cat/en/condensation-cube (besucht am 31.08.2012).

[38] Geoff Cox, Alex McLean und Adrian Ward. *The Aesthetics of Generative Code*. URL: http://generative.net/papers/aesthetics/ (besucht am 06.10.2012).

[39] Martin Deppe. *Der ewige Krieg*. 2012. URL: http://www.gamestar.de/spiele/civilization-5-gods-kings/artikel/10_jahre_partie_in_civilization_2,47855,2569947.html (besucht am 15.07.2012).

[40] Reinhard Döhl. *Exkurs über Aleatorik*. URL: http://www.stuttgarter-schule.de/aleatori.htm (besucht am 22.05.2012).

[41] Thomas Dreher. *Von „Radical Software“ zum Netzaktivismus*. 2007. URL: http://iasl.uni-muenchen.de/links/NARS.html (besucht am 15.09.2012).

[42] Karlheinz Essl. *Plädoyer für „Das Offene Kunstwerk“*. 1996. URL: http://www.essl.at/bibliogr/ok.html (besucht am 25.07.2012).

[43] Philip Galanter. *About*. 2010. URL: http://philipgalanter.com/about/ (besucht am 24.07.2012).

[44] Philip Galanter. *generative art and rules-based art*. 2006. URL: http://66.49.250.143/content/archives/journal03/galanter/generative_art_and_rules_based_art.pdf (besucht am 24.05.2012).

[45] Philip Galanter. *What is Generative Art?: Complexity Theory as a Context for Art Theory*. 2003. URL: http://philipgalanter.com/downloads/ga2003_what_is_genart.pdf (besucht am 03.08.2012).

[46] *Game of Life*. URL: http://de.wikipedia.org/wiki/Game_of_Life (besucht am 06.10.2012).

[47] *Hans Richter*. URL: http://de.wikipedia.org/wiki/Hans_Richter_(Dadaist) (besucht am 06.10.2012).

[48] *Happening*. URL: http://de.wikipedia.org/wiki/Happening (besucht am 06.10.2012).

[49] Tjark Ihmels und Julia Riedel. *Die Methodik der generativen Kunst*. URL: http://www.medienkunstnetz.de/themen/generative_tools/generative_art/scroll/ (besucht am 22.05.2012).

[50] Anish Kapoor. *Non-Object (Spire)*. 2007. URL: http://www.anishkapoor.com/121/Non-Object-%28Spire%29.html (besucht am 06.10.2012).

[51] Ken Knowlton und Leon Harmon. *Studies in Perception #1*. 1966. URL: http://www.knowltonmosaics.com/pages/HKnewd.htm (besucht am 06.10.2012).

[52] *Koch snowflake*. URL: http://en.wikipedia.org/wiki/Koch_snowflake (besucht am 06.10.2012).

[53] *Konzeptkunst*. URL: http://de.wikipedia.org/wiki/Konzeptkunst (besucht am 06.10.2012).

[54] *Künstliche Intelligenz*. URL: http://de.wikipedia.org/wiki/K%C3%BCnstliche_Intelligenz#Turing-Test (besucht am 06.10.2012).

[55] Golan Levin. *Interview by Alexandra Nemerov*. 2007. URL: http://flong.com/texts/interviews/interview_nemerov/ (besucht am 24.05.2012).

[56] Golan Levin. *Interview by Carlo Zanni for CIAC Magazine*. 2004. URL: http://flong.com/texts/interviews/interview_ciac/ (besucht am 24.05.2012).

[57] Sol LeWitt. *Proposal for a Wall Drawing, Information Show*. 1970. URL: http://ideasandimages.files.wordpress.com/2010/05/proposal-for-a-wall-drawing-information-show.jpg (besucht am 01.06.2012).

[58] *Lindenmayer-System*. URL: http://de.wikipedia.org/wiki/Lindenmayer-System (besucht am 06.10.2012).

[59] *Lisp*. URL: http://de.wikipedia.org/wiki/Lisp (besucht am 06.10.2012).

[60] Elise Malmberg. *Joshua Davis: Infinitely Interesting*. URL: http://www.apple.com/pro/profiles/joshuadavis/ (besucht am 24.05.2012).

[61] Manfred Mohr. *selection from early algorithmic experiments 1969-1973*. URL: http://www.emohr.com/two_1969.html (besucht am 23.05.2012).

[62] James Moore. *I've been playing the same game of Civilization II for almost 10 years. This is the result.* 2012. URL: http://www.reddit.com/r/gaming/comments/uxpil/ive_been_playing_the_same_game_of_civilization_ii/ (besucht am 15.07.2012).

[63] Ingo Mörth, Cornelia Hochmayr und Katja Kwastek. *rezeption interaktiver kunst: eine analyse der produktionsvorstellungen und rezeptionsrealität von interaktiver kunst im kontext der ars electronica 2007*. 2008. URL: http://www.kuwi.jku.at/interaktive_kunst.pdf (besucht am 23.07.2012).

[64] Rob Myers. *generative art is...* 2006. URL: http://66.49.250.143/content/archives/journal03/myers/generative_art_is.pdf (besucht am 24.05.2012).

[65] *Op-Art*. URL: http://de.wikipedia.org/wiki/Op-Art (besucht am 06.10.2012).

[66] Craig Reynolds. *Boids*. 2007. URL: http://www.red3d.com/cwr/boids/ (besucht am 06.10.2012).

[67] *Screenshots*. URL: http://vvvv.org/screenshots (besucht am 06.10.2012).

[68] Jeffrey Thompson. *Alex Dragulescu's "Malwarez" series*. 2010. URL: http://www.jeffreythompson.org/blog/2010/07/30/alex-dragulescus-malwarez-series/ (besucht am 10.09.2012).

[69] *Turing-Test*. URL: http://de.wikipedia.org/wiki/Turing-Test (besucht am 06.10.2012).

[70] Roman Verostko. *The Algorists*. URL: http://verostko.com/algorist.html (besucht am 23.05.2012).

[71] Matt Ward. *The Poetics Of Coding*. 2010. URL: http://coding.smashingmagazine.com/2010/05/05/the-poetics-of-coding/ (besucht am 11.09.2012).

[72] Martin Wattenberg. *Arc Diagrams: Visualizing Structure in Strings*. 2002. URL: http://www.research.ibm.com/visual/papers/arc-diagrams.pdf (besucht am 05.06.2012).

[73] Martin Wattenberg. *The method behind The Shape of Song*. URL: http://www.turbulence.org/Works/song/method/method.html (besucht am 05.06.2012).

[74] Marius Watz. *Closed systems: Generative art and Software Abstraction*. 2010. URL: http://mariuswatz.com/2012/03/30/closed-systems-generative-art-and-software-abstraction/ (besucht am 06.10.2012).

[75] Marius Watz. *fragments on generative art*. 2006. URL: http://66.49.250.143/content/archives/journal03/watz/fragments_on_generative_art.pdf (besucht am 24.05.2012).

[76] Peter Wegner. *The Paradigm Shift from Algorithms to Interaction*. 1996. URL: http://www.cs.brown.edu/people/pw/papers/ficacm.ps (besucht am 03.08.2012).

[77] *Zellulärer Automat*. URL: http://de.wikipedia.org/wiki/Zellul%C3%A4rer_Automat (besucht am 06.10.2012).