

Generative Typografie.

**Eine Untersuchung generativer Gestaltungsstrategien
hinsichtlich der Erzeugung von Displayschriften mit VVVV.**

Diplomarbeit zur Erlangung des akademischen Grades „Magister (FH)“

Verfasser: Philipp Steinweber

Vorgelegt am FH-Studiengang MultiMediaArt, Fachhochschule Salzburg

Begutachtet durch:

DI Brigitte Jellinek

Christian Süß

Frankfurt, 07. September 2008

Danksagung

Mimi

Mama

Strukt, Thomas, Tanja, Wolfi, Andi, Iris, Robi

Brigitte, Christian

Kathi

Maria, Markus

Sanch, Tebjan, Woei, David, Björn, Ingolf

Joreg, Gregsn, Sebastian, Max

Ricard, Benedikt, Florian

Eidesstattliche Erklärung

Hiermit versichere ich, Philipp Steinweber, geboren am 09.12.1983 in Rosenheim, dass ich die Grundsätze wissenschaftlichen Arbeitens nach bestem Wissen und Gewissen eingehalten habe und die vorliegende Diplomarbeit von mir selbstständig verfasst wurde. Zur Erstellung wurden von mir keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Ich versichere, dass ich die Diplomarbeit weder im In- noch Ausland bisher in irgendeiner Form als Prüfungsarbeit vorgelegt habe und dass diese Arbeit mit der den BegutachterInnen vorgelegten Arbeit übereinstimmt.

Frankfurt, den 07.09.2008

Philipp Steinweber

Matrikelnummer: 0410055046

Kurzfassung der Arbeit

Vor- und Zuname: Philipp STEINWEBER
Institution: FH Salzburg
Studiengang: MultiMediaArt
Titel der Diplomarbeit: Generative Typografie.
Eine Untersuchung generativer Gestaltungsstrategien hinsichtlich der Erzeugung von Displayschriften mit VVVV.
Begutachterin (1): DI Brigitte Jellinek
Begutachter (2): Christian Süß

Schlagwörter

1. Schlagwort: generative Gestaltung
2. Schlagwort: Typografie
3. Schlagwort: VVVV

In dieser Arbeit werden generative Gestaltungsmethoden hinsichtlich der Erzeugung von Displayschriften mit VVVV untersucht.

Generative Gestaltung ist eine spezifische Herangehensweise an den Gestaltungsprozess, welche sich von heute gängigen „What you see is what you get“-Methoden unterscheidet. GestalterInnen übersetzen ihre Vorstellung von einem visuellen Produkt in Algorithmen – die Sprache des Mediums Computer. Durch diese sprachliche Annäherung ist es möglich, Computern andere, dem Medium eher entsprechende Ergebnisse zu entlocken. Getreu der Attitüde von Ivan E. Sutherland: „*It is only worthwhile to make drawings on the computer if you get something more out of the drawing than just a drawing*“ (Sutherland 1963, 507).

Die Arbeit durchleuchtet den Prozess, den GestalterInnen beim generativen Gestalten durchlaufen. Dieser unterscheidet sich, durch die Übersetzung in Maschinensprache, entscheidend vom herkömmlichen Gestaltungsprozess. Signifikante Parallelen sind hierbei in der frühen Computerkunst zu finden.

Um der generativen Erzeugung von Displayschriften näher zu kommen wird Schrift hinsichtlich ihres digitalen Darstellungsprozesses

untersucht. Verfahren generativer Gestaltung können an verschiedenen Punkten dieses Prozesses eingreifen und die Darstellung, entsprechend den Wünschen der GestalterInnen, algorithmisch verändern.

Im weiteren Verlauf werden konkret einige Methoden generativer Gestaltung herausgegriffen, und hinsichtlich ihrer Möglichkeiten Displayschriften zu generieren und zu beeinflussen untersucht. Somit entsteht ein Grundstock an Möglichkeiten, dieses Vorhaben in die Praxis zu überführen.

Mit dem Werk *touch.txt* demonstriert der Autor dies Anhand von Multi-touch-Anwendungen, und lässt BenutzerInnen die so erschaffenen Bildklassen eigenhändig erforschen.

Abstract

This present work analyzes generative attempts for creating display typefaces.

Generative Design is a specific method for creating visual content, which completely differs from currently conventional “What You See Is What You Get” operations. The designer has to translate his imagination of the visual product into algorithms, the language of the medium computer. This linguistic approach to the medium enables creating a different kind of visual results. Proper to Evan E. Sutherlands principles: *“It is only worthwhile to make drawings on the computer if you get something more out of the drawing than just a drawing”* (Sutherland 1963, 507).

The processes the designer passes this way, differ decisively from conventional methods. The most common parallels can be found in the early computer graphics.

For approaching the topic of generative creation of display faces, font is being examined in terms of its computational process. Methods of generative design are able to tweak this process algorithmically at specific points, for implementing the designers vision.

Further on, specific methods of Generative Design are being singled out and analyzed, in terms of their possibilities for creating and influencing display faces. This leads to a basis of possibilities, which allows implementing those approaches.

That's what the piece touch.txt does in several multitouch applications. Users are so able to investigate the evolving image classes.

Inhaltsverzeichnis

Einleitung	1
1. Generative Gestaltungsstrategien	6
1.1 Begriffsklärung	7
1.1.1 Was ist generative Gestaltung?	7
1.1.1.1 Generativ versus Herkömmlich	8
1.1.1.2 Algorithmen	9
1.1.1.3 Der Begriff der Bildklasse	10
1.1.1.4 Mehrwert der Methoden	11
1.1.1.5 Definition	12
1.1.2 Vom Kreieren von Bildklassen	13
1.2 Entstehung	17
1.2.1 Frühe formale Vorreiter: Konstruktivismus, DeStijl und Bauhaus	17
1.2.2 Kulturell-künstlerische Auseinandersetzungen mit Apparaten.	18
1.2.2.1 Autorenschaft in der Maschinenkunst	19
1.2.2.2 Der Parameterbegriff in der Maschinenkunst	20
1.2.2.3 Kunst trifft Wissenschaft: OpArt	21
1.2.3 Digitalisierung	21
1.2.3.1 Anfänge der Computergrafik	21
1.2.3.2 Der Weg zur aktuellen Computergrafik	28
1.3 Werkzeuge	30
1.3.1 Scriptographer	30
1.3.2 VVVV	33
1.3.2.1 Konzept der visuellen Programmierung in VVVV	34
1.3.2.2 Benutzeroberfläche und Bedienung	36
1.3.2.3 generatives Gestalten in VVVV	38
1.4 Zusammenfassung	44
2. Typografie	46
2.1 Einleitung	47
2.2 Bedeutung von Typografie in dieser Arbeit	48
2.3 Digitale Typografie	52
2.3.1 Schrift in Windows – von der Schriftdatei zur Bildschirmausgabe	53

2.3.2	Schrift in VVVV	55
2.3.2.1	Schrift im GDI Renderer	56
2.3.2.2	Typospread	57
2.3.2.3	Schrift als 3d Objekt	59
3.	Methoden	65
3.1	Einleitung	66
3.2	Zufall	68
3.2.1	Hinführung	68
3.2.2	Das Prinzip Pseudozufall	69
3.2.3	Zufallsgeneratoren in VVVV	70
3.2.4	Typografische Anwendungen	71
3.3	Partikelsysteme	79
3.3.1	Hinführung	79
3.3.2	Funktion von Partikelsystemen	80
3.3.3	Partikelsysteme in VVVV	81
3.3.3.1	Partikelerzeugung mit Hilfe von Spreads	82
3.3.3.2	Partikel Plugin in VVVV	83
3.3.4	Typografische Anwendungen	85
3.3.4.1	Partikel in ihrer Reinform: Punkte	85
3.3.4.2	Partikel kreieren Linien	87
3.4.	Attraktoren	92
3.4.1	Attraktoren in VVVV	92
3.4.2	Typografische Anwendungen	93
4.	Schluss	100
	Literaturverzeichnis	105
	Abbildungsverzeichnis	112
	Glossar	115
	Anhang: Werkdokumentation	A1

Abkürzungsverzeichnis

bzw	beziehungsweise
dh	das heißt
Hg	Herausgeber
unpag	unpaginiert
uvm	und viele(s) mehr
vgl	vergleiche
zB	zum Beispiel
zit n	zitiert nach

Einleitung

Diese Arbeit hat das Ziel generative Gestaltungsmethoden für die Erstellung von Displayschriften zu untersuchen.

Generative Gestaltung beschreibt eine Herangehensweise an den Prozess Gestaltung, welche sich von den in heutiger Zeit gängigsten Methoden unterscheidet. Das Werkzeug Computer bleibt bestehen. Die Programmebene an der die GestalterInnen eingreifen, um ihre Ideen umzusetzen ist jedoch eine andere, als die, in welcher sich die Software aufhält, die das „What you see is what you get“-Prinzip verfolgt.

Generative Gestaltung erlebt zum jetzigen Zeitpunkt einen Aufwärts-trend. Die technischen Möglichkeiten der Programmierumgebungen VVVV und Processing wachsen, was die Benutzerzahlen kontinuierlich steigen lässt.

Einen Beleg dafür liefert die Tatsache des Einzugs in die Popmusik: die Band Radiohead veröffentlichte am 13. Juli 2008 ihr aktuelles Musik-video („House of Cards“) auf Google Code. Das Video besteht nicht aus Bildern, die mit einer Kamera aufgezeichnet wurden, sondern aus Ergebnissen von verschiedenen 3d Scannern: aus Daten (vgl Radiohead 2008a).

Diese stehen frei zum Download zur Verfügung, mit der Aufforderung, sie zu *remixen*. Die BenutzerInnen werden motiviert eigene Visualisie-rungen der Daten zu gestalten. Ein Unterfangen, welches in erster Linie mit generativen Gestaltungsmethoden umgesetzt werden kann – und umgesetzt wird, wie die zahlreichen Einreichungen der zugehörigen YouTube Gruppe¹ zeigen.

¹ <http://www.youtube.com/group/houseofcards> – aufgerufen am 13.08.08, 10 Uhr

Das Themenfeld der generativen Gestaltung wird in der vorliegenden Arbeit aus den Anfängen der Computerkunst hergeleitet: einem Zeitpunkt der Geschichte, an dem erstmalig die Möglichkeit bestand, gestalterische Produkte zu *programmieren*. Dies führte zum Konzept der heutigen, generativen Gestaltung:

„*Dem traditionellen Künstler gehe es um die eine Zeichnung. Der Programmierer beschreibt das Schema aller Zeichnungen.*“
(*Nierhoff-Wielk 2007, 23*)

Die Kernliteratur hierfür bildet der Ausstellungskatalog der Kunsthalle Bremen zur Ausstellung „Ex Machina“ (siehe *Nierhoff-Wielk 2007*), der die Entstehungsgeschichte der Computerkunst veranschaulicht. In verschiedenen Beiträgen wird die Sicht der Künstler aufgezeigt, und ihrer Motivation, die Geräte zweckzuentfremden, nachgegangen.

Im nächsten Schritt werden die Methoden in die Gegenwart transferiert, wobei die entscheidende Literatur hierbei das Buch *Code@Art* bildet (siehe *Trogemann/Viehoff 2005*).

Der Themenbereich wird durch die Schriften des Gestalters John Maeda (siehe *Maeda 1999*) in den Designdiskurs gebracht, der eine *neue* Definition für generative Gestaltung fordert, die voraussetzen soll, dass die angewandten Verfahren ästhetisch gerechtfertigt sind (vgl *Maeda 199, 175*).

Im Folgenden werden spezifische Methoden dieser generativen Herangehensweise untersucht, mit dem Ziel Displayschriften generativ zu erzeugen bzw zu beeinflussen.

Displayschriften sind ein Bereich der Typografie, der für experimentelle und avantgardistische Auswüchse bekannt ist und deshalb ein großes Experimentierfeld für die Untersuchungen bietet.

Displayschriften sind hauptsächlich in der Plakat-, Flyer und Buchgestaltung, aber auch im Webdesign und bei interaktiven Anwendungen zu finden. Ihr Ziel ist häufig auf illustrative Art Stimmungen zu kommunizieren, worunter auch die Lesbarkeit leiden darf.

Die theoretische typografische Basis ist primär auf dem Sammelband *Type One* des Gestalten Verlags (siehe *Bilz 2004*) begründet, der sich zeitgenössischen Strömungen der Schrifterstellung widmet. Abgerundet werden die Erkenntnisse durch allgemeinere Typografie-Theorie aus dem Typolexikon (siehe *Beinert 2008a* und *Beinert 2008b*).

Generative Ansätze existieren in diesen Gebiet bereits, dennoch ist das gestalterische Innovationspotenzial als sehr hoch einzustufen, da die technische Entwicklung der Programmierumgebungen für generative Gestaltung momentan den Ideen der GestalterInnen vorausseilt. Ziel dieser Arbeit ist es, die Themen, generative Gestaltung und Typografie, zusammenzuführen.

Da der Begriff der Typografie wird in dieser Arbeit lediglich unter dem Aspekt der Gestaltung von Schriften betrachtet. Noch konkreter wird sich der Gestaltung von Displayschriften gewidmet, da diese den voraussichtlich notwendigen, unkonventionellen Rahmen am besten stecken.

Bezüglich generativer Gestaltung werden spezifische Methoden aus dem Pool von Möglichkeiten herausgegriffen und in ihrer Tauglichkeit untersucht, Displayschriften zu generieren.

Darüberhinaus beschränkt sich die Herleitung auf die Software VVVV.

Daraus ergeben sich folgende Forschungsfragen:

1. Eignet sich generative Gestaltung für die Erzeugung von Displayschriften?

falls ja:

2. Welche generativen Gestaltungsmethoden von VVVV eignen sich für die Erzeugung von Displayschriften?

Um dies zu beantworten widmet sich das erste Kapitel dem Begriff der generativen Gestaltung. Dabei gilt es anfangs in erster Linie den Begriff zu definieren, und die Bedeutung für den Gestaltungsprozess zu erörtern.

Im Anschluss wird die Herkunft dieser Vorgehensweise erläutert, um ein größeres Verständnis bei den LeserInnen zu erreichen, und den Bogen hin zu aktuellen Entwicklungen zu spannen.

Diese werden auch im weiteren Verlauf verdeutlicht, wenn die Möglichkeiten und spezifische Verfahren der Programmierumgebung VVVV erläutert wird.

Somit ist am Ende dieses ersten Kapitels ein Überblick über generatives Gestalten erreicht, und der Begriff ist in seiner Bedeutung analysiert, und bereits ansatzweise in die Praxis überführt worden.

Im zweiten Kapitel wird das zweite große Thema behandelt, die Typografie.

Nach Begriffsklärungen wird in die digitale Typografie übergegangen und erläutert, welche Schritte notwendig sind, damit an heutigen Windows Computern Schrift am Bildschirm dargestellt werden kann. Dies ist Vorbereitung für spätere generative Eingriffe, da diese an verschiedenen Stufen des Prozesses ansetzen können.

Im letzten Abschnitt werden die verschiedenen Schriftfunktionen von VVVV vorgestellt, und untersucht, inwiefern diese sich für generative Eingriffe eignen, und an welcher Stelle der Schriftdarstellung sie eingreifen.

Somit ist der Weg geebnet, die beiden weitreichenden Begriffe zusammenzuführen. Dies ist die Aufgabe des dritten Kapitels.

Spezifische Methoden generativer Gestaltung werden ausgewählt und auf ihre Tauglichkeit für die Erzeugung von Displayschriften untersucht. Als Abschluss werden die gewonnenen Erkenntnisse zusammengefasst, und kritisch diskutiert.

Aus Gründen der Lesbarkeit verzichtet der Autor auf die Punktierung bei Abkürzungen.

Darüberhinaus werden Fehler in Zitaten, die Literatur vor der Rechtschreibreform entspringen, und die nach alter Rechtschreibung keine Fehler wären, nicht mit (*sic!*) gekennzeichnet.

Der Autor verfasst diese Arbeit im Rahmen eines qualitativen Forschungsdesigns. Das bedeutet für die Herangehensweise „*Einzelphänomene werden gesondert behandelt, die Struktur wird analysiert, der Aussagewert interpretiert und in einen Gesamtkontext eingebettet*“ (MANFÉ/MAIRITSCH 2008, 12).

Zusätzlich wird die Werkoption gewählt, um theoretisch eingeleitete Gedanken auch entsprechend visualisieren zu können. Dies ist für die Arbeit notwendig, da sie ohne diese Zirkularität nicht funktionieren könnte. Die Forschungsfrage ist so spezifisch, dass keine aktuelle Literatur sie alleine beantworten könnte. Vielmehr baut diese Arbeit einen theoretischen Grundstock auf, um im weiteren Verlauf Theorie und Praxis gemeinsam zu gestalten.

1. **Generative
Gestaltungsstrategien**

1.1 Begriffsklärung

In der folgenden Arbeit geht es um eine spezifische Methode für die Erzeugung von Formen, Farben, Kompositionen; es geht um generative Gestaltung.

In diesem ersten Kapitel wird erläutert, worin sich diese Herangehensweise von anderen unterscheidet, welche verschiedenen Schritte in der Entwicklung notwendig waren um technische sowie kulturelle Rahmenbedingungen zu schaffen, und ein Überblick über mögliche Anwendungen und entsprechende Werkzeuge geboten.

Anfangs ist es notwendig, einige Begriffe zu definieren.

1.1.1 Was ist generative Gestaltung?

Generative Gestaltung steht im Mittelpunkt dieser Arbeit und wird für eine Annäherung an den Gesamtbegriff anfangs wörtlich separiert betrachtet:

„Gestaltung ist ein kreativer Schaffensprozess, bei welchem durch die Arbeit des Gestaltenden eine Sache (ein materielles Objekt, eine Struktur, ein Prozess, ein Gedankengut etc.) verändert wird, d.h. erstellt, modifiziert oder entwickelt wird und dadurch eine bestimmte Form oder ein bestimmtes Erscheinungsbild verliehen bekommt oder annimmt.“

(Wikipedia 2008a).

Der Begriff Gestaltung schließt den Begriff des Prozesses bereits ein – beim Ausdruck *generative Gestaltung* konkretisiert das Wort *generativ* die Beschaffenheit des Prozesses.

Generativ beschreibt den Prozess, den der Gestalter zurücklegen muss um seine Idee in ein gestaltetes Produkt umzuwandeln. Der generative Prozess bedient sich Apparaten, Maschinen oder Programmen, deren Benutzung das Gestaltungsprodukt entscheidend prägt.

In der vorliegenden Arbeit werden ausschließlich visuelle Gestaltungsergebnisse betrachtet. Diese werden im folgenden auch als Grafik bezeichnet.

Darüberhinaus beschränkt sich die Arbeit auf die Generierung und Veränderung von Grundformen und Buchstaben, und klammert somit

die Verwendung von importiertem Material (Bilder, Videos) aus. Diese Einschränkung ermöglicht eine bessere Abgrenzung zu sonstigen Gestaltungsmethoden, und wird auch in vergleichbaren Herleitungen dieser Art vorgenommen (vgl Hitthaler 2005, 24).

1.1.1.1 Generativ versus Herkömmlich

Die historische Herleitung ausgenommen, befasst sich diese Arbeit mit generativer Gestaltung am Computer. Somit ist zu klären, ob und wie sich der generative Prozess von dem herkömmlichem Gestaltungsprozess am Computer unterscheidet. Die heute gängige Art der Grafi-kerstellung am Computer nutzt intuitive Eingabemethoden, die den GestalterInnen Übersetzungsprozesse ersparen will. Soll ein Rechteck gezeichnet werden, wird dies *direkt* mit der Maus auf den Bildschirm gezeichnet. Dieses Methoden sind auch unter dem Begriff WYSIWYG (What you see is what you get) bekannt. Die populären Programme Photoshop und Illustrator von Adobe sind Beispiele heutiger WYSIWYG Software.

Der generative Prozess hingegen funktioniert anders. Eine Gemeinsamkeit allerdings ist, dass sich beide Verfahren Programme zu Nutzen machen:

„Nun kann man sich die Entstehungsgeschichte jeder Graphik mindestens im nachhinein als nach einem wohldefinierten Programm ablaufen vorstellen, da auch im handwerklichen Vollzug Richtung und Gliederung zu erkennen und Etappen zu unterscheiden sind. Im Rahmen generativer Graphik jedoch ist das Programm mit seiner Struktur die erste und bewußt konstituierte Komponente des ästhetischen Erzeugungsprozesses, derart, daß von der Struktur der erzeugten Graphik jederzeit auf die dokumentierte Struktur des erzeugten Programms zurückgefragt werden kann.“

(Nees 1969, 25)

Dieses Zitat von Georg Nees kommt aus einer Zeit vor dem Durchbruch des WYSIWIG Prinzips. Aufgrund des hohen Intuitionsgrads, den diese Programme erlauben, können die WYSIWYG Verfahren aber mit den von Nees beschriebenen, handwerklichen Verfahren verglichen werden. Die gestalterische Qualität ist dabei davon abhängig, wie gut die Werkzeuge (Maus) geführt werden – vergleichbar mit dem Pinsel aus Zeiten der Malerei.

Um einer Definition des generativen Gestaltungsprozesses näher zu

kommen, kann man aus Nees Zitat mitnehmen, dass sich Strukturen der Grafik, und Struktur des Programms näher sind, als in WYSIWYG Anwendungen.

Die Position der GestalterInnen verändert sich so im Vergleich zu der Herkömmlichen: statt sich in vorgegebenen Programmstrukturen zu bewegen, setzen GestalterInnen die sich generativen Verfahren bedienen eine Ebene weiter oben an: sie gestalten das Programm selbst. Das hat zur Folge, dass das Regelwerk der Software nicht als Konstante, in deren Rahmen man sich zu bewegen hat, betrachtet werden muss, sondern als dynamisch flexibler Baukasten.

Generative GestalterInnen sind auch ProgrammiererInnen. Programmieren ist hierbei ein „*zeichenbasiertes Verfahren, um die Tätigkeiten von Maschinen festzulegen*“ (Trogemann/Viehoff 2005, 38).

1.1.1.2 Algorithmen

Durch das Programmieren haben GestalterInnen im Vergleich zu herkömmlichen Verfahren Kontrolle über Algorithmen, die die Grafik beschreiben.

„*Algorithmen sind nach heutiger Auffassung endliche, schrittweise Verfahren zur Berechnung gesuchter Größen aus gegebenen Größen*“ (Trogemann/Viehoff 2005, 180). In dieser Arbeit wird fortan der Begriff Parameter statt dem der Größe verwendet, da dieser Begriff in der gestalterischen Anwendung gängiger ist.

Der Begriff Algorithmus bezeichnet ursprünglich Verfahren und deren Anwendung in der Mathematik, z.B. Lösungsverfahren für bestimmte Gleichungen – heute ist die Schirmwissenschaft dieses Begriffs die Informatik.

Laut Alfred Schreiber muss ein Regelwerk folgende Kriterien erfüllen, um als Algorithmus angesehen werden zu können:

- „I. *Diskretheit. Ein Algorithmus besteht aus einer Folge von Schritten.*
- II. *Determiniertheit. Bei gleichen Startbedingungen erzeugt er stets dasselbe Endergebnis.*
- III. *Eindeutigkeit. Nach jedem Schritt lässt er sich auf höchstens eine Art fortsetzen.*
- IV. *Endlichkeit. Er endet nach endlich vielen Schritten.*“
(Schreiber 2000)

1.1.1.3 Der Begriff der Bildklasse

Die Verfahren, die hier betrachtet werden, bedienen sich Algorithmen. Ein generatives Werk besteht aus einer Vielzahl von Teilalgorithmen, aus einem System mehrerer kleinen Regeln, die im Gesamten ein komplexes *generatives System* bilden.

Für das Verständnis von grafischen generativen Werken ist entscheidend, dass es sich nicht um ein Bild handeln kann, sondern um eine Bildklasse. Ein generatives System beinhaltet nicht ein einzelnes Bild, sondern ein Schema vieler Bilder (vgl Nierhoff/Wielk 2007, 22).

Bei einer gedruckten Grafik betrachtet man ein Bild, das aus einem generativen System herausgenommen wurde. Unabdingbar ist, dass dieses eine Bild noch eine Vielzahl Verwandter hat, die dem selben generativen System jederzeit entspringen können, sofern die GestalterIn das will.

Verständlicher wird dies, wenn man den Begriff *generatives System* in anderen Wissenschaften betrachtet, wie in der Biologie:

„So I thought we would delve into generative systems across a broad range of topics here, you know kind of one of the pre-eminent ones is biology. And this is one that we're very familiar with. Richard Dawkins, the biologist, estimated that the typical willow seed actually all it contains is about 800k of data, which is enough to fit on one of the old floppy disks. Which is really amazing, when you imagine all of the trillions of atoms and all the complexity in a willow tree, that the genome of it compresses down to that small is very powerful ratio there.“

(Wright 2006, 4)

Das generative System befindet sich in diesem Beispiel im Samen des Baums¹, woraus eine Vielzahl von Bäumen entstehen kann. Aufgrund der äußeren Bedingungen (Klima, Witterung uvm) entsteht aber genau ein Baum, der von diesen Umständen geformt wird. Diese Startbedingungen, also die Inputs des Systems, werden fortan als Parameter bezeichnet.

Eine GestalterIn, die generative Verfahren anwendet, kann diese äußeren Parameter in ihrem generativen Systemen nach belieben beeinflussen und auswählen, welcher Baum entstehen soll.

¹ Es handelt sich hier um ein genetisches und nicht um ein algorithmisches Regelwerk

1.1.1.4 Mehrwert der Methoden

Bevor nun zu einer, für diese Arbeit gültige Definition gekommen werden kann, liegt es im Interesse des Autors noch eine zusätzliche Einschränkung vorzunehmen.

Generative Gestaltungsmethoden sind theoretisch in der Lage sämtliche Eigenschaften von WYSIWYG Programmen zu übernehmen, und exakt gleiche Ergebnisse zu erzielen. Dies ist jedoch nicht erstrebenswert – oder wie Ivan E. Sutherland sagte: „*It is only worthwhile to make drawings on the computer if you get something more out of the drawing than just a drawing*“ (Sutherland 1963, 507).

Generative Gestaltungsmethoden anzuwenden, ermöglicht Werke zu erzeugen, deren Erscheinung und Komplexität mit anderen Methoden nicht erreicht werden kann. Die Intention, diese Art von Werken zu erzeugen, soll als Voraussetzung mit in die Definition generativer Gestaltungsstrategien fließen.

„(...) drawing a stroke with a pen is no different from drawing a stroke with a mouse. The real challenge is to discover the intrinsic properties of the new medium and to find out how the stroke you ‘draw’ via computation is one you could never draw, or even imagine, without computation.“

(Maeda 1999, 175)

Im weiteren Verlauf werden aus Verständnisgründen oft formal triviale Beispiele, wie das Zeichnen eines Rechteckes betrachtet. Diese sollen den Prozess generativer Gestaltungsstrategien verdeutlichen – echte generative Gestaltung bedarf nach dem Zeichnen des Rechteckes jedoch zusätzliche Schritte, um den nötigen gestalterischen Mehrwert aufzubringen.

1.1.1.5 Definition

Aus obigen Erläuterungen ableitend, wird generative Gestaltung für diese Arbeit wie folgt definiert:

Wenn man von generativen Gestaltungsstrategien spricht, meint man damit, dass die GestalterIn mit Hilfe von Algorithmen ein generatives System kreiert, welches das Ziel hat visuell ansprechende Bilder hervorzu bringen, deren Ästhetik und/oder Komplexität das Verfahren rechtfertigen.

Die GestalterIn ist damit einerseits ErzeugerIn eines Regelwerks, das unendlich viele visuelle Ergebnisse hervorbringen kann, und andererseits KuratorIn die auswählt, welche Ergebnisse als Teil des Produktes System, als ästhetische Repräsentanten des Systems dienen sollen.

Mit dieser Definition wird der erste Abschnitt dieser Arbeit abgeschlossen. Das nächste Kapitel widmet sich dem Prozess, den generative Gestaltungsstrategien für die GestalterInnen bedeuten.

1.1.2 Vom Kreieren von Bildklassen

*„Ich bin der Musikant mit Taschenrechner in der Hand.
Ich addiere,
und subtrahiere,
kontrolliere,
und komponiere.
Und wenn ich diese Taste drück‘
spielt er ein kleines Musikstück.“*

Kraftwerk – Taschenrechner (Album Computerwelt 1986)

Bereits im Jahre 1986 huldigt die Band Kraftwerk die Maschine Computer als kreatives Werkzeug, indem sie den Gestaltungsprozess überspitzt und offen darlegt.

Der Prozess ist bei generativer Gestaltung der entscheidende Unterschied zur WYSIWYG Gestaltung, weshalb ihm an dieser Stelle noch ein eigener Abschnitt gewidmet sei, bevor die geschichtliche Herkunft näher ausgeführt wird. Aus dem Prozess ergibt sich die ungewöhnliche Form des Produkts: die Bildklasse. Zu verstehen was dies für den Gestalter bedeutet und welche Art der Abstraktion und Übersetzung bei der Herstellung notwendig ist, ist unabdingbar für das Verständnis der späteren Kapitel.

Im ersten Abschnitt wurde bereits erläutert, dass sich generative Gestaltung Maschinen, Apparate, und in den meisten Fällen den Computer zu Nutze macht. Der Prozess, den die GestalterIn zurücklegen muss, ist bei allen drei Varianten sehr ähnlich, weshalb die Prinzipien der Mechanisierung und der mechanistischen Weltauffassung in diesen Zusammenhang gebracht werden können. Mechanisierung kann als das Ergebnis einer rationalen Einstellung zur Welt betrachtet werden (vgl Trogemann/Viehoff 2005, 9), weshalb einige Übersetzungs- und Abstraktionsverfahren nötig sind, um eine irrationale Idee auf generative Weise umzusetzen:

„Das Zusammenwirken der drei Prinzipien – Abstraktion, Formalisierung und Mechanisierung – bilden zusammen den eigentlichen Kern der mechanistischen Weltauffassung.“

(Trogemann/Viehoff 2005, 32)

Gestaltung generativ zu erzeugen heißt, diesen mechanistischen Prozess zu durchlaufen. Im Vergleich zur herkömmlichen Gestaltung ist diese phasenweise Separierung des Gestaltungsprozesses beim generativen Gestalten sehr ausgeprägt.

An dieser Stelle kann der offengelegte Prozess der Band Kraftwerk aus obigem Zitat herbeigezogen werden. Eine Analyse dieser Prozessaufschlüsselung ergibt, dass sich alle drei Prinzipien laut der Definition von Trogemann bei Kraftwerk finden lassen.

Anfangs findet die Abstraktion und die Formalisierung statt: „*Ich bin der Musikant mit Taschenrechner in der Hand. Ich addiere, und subtrahiere, kontrolliere und komponiere*“ (Kraftwerk 1986).

Abstraktion bedeutet gedankliche Zerlegung eines Vorgangs in seine Teilprozesse (vgl Trogemann/Viehoff 2005, 31). Kraftwerk zerlegen ihr Musikstück in einzelne, rationale Pakete, um sie im darauf folgenden Schritt, der Formalisierung, in eine allgemeingültige Zeichensprache zu übersetzen. Dies passiert im Kraftwerk Beispiel durch die Eingabe am Taschenrechner. Die Musik wird in mathematische Prinzipien zerlegt, und anschließend in deren gültiger Zeichensprache formuliert. „*Abstraktion und Formalisierung sind damit immer Analyse und Synthese zugleich*“ (Trogemann/Viehoff 2005, 31).

Der dritte Schritt der Mechanisierung befindet sich im letzten Satz: „*Und wenn ich diese Taste drück', spielt er ein kleines Musikstück*“ (Kraftwerk 1986). Aufgrund der vorangegangenen Übersetzung ist es möglich für eine Maschine, das eingegebene Werk (Musikstück) zu berechnen. Dieser Vorgang ist wiederholbar, die Algorithmen können von der Maschine beliebig oft ausgeführt werden.

Bezieht man diesen Prozess auf ein Beispiel aus der gestalterischen Praxis, kommt man schnell zu dem Schluss, dass generative Gestaltung spezifische Anwendungsgebiete braucht. Im Vergleich zum WYSIWYG Prinzip, wirkt es äußerst kompliziert, die aufgezählten Schritte durchlaufen zu müssen, um ein Rechteck zu zeichnen. WYSIWYG Anwendungen erlauben ein direktes Zeichnen des Objekts, wobei die Parameter wie Höhe und Breite mit dem Eingabegerät bestimmt werden.

Um ein Rechteck generativ zu erstellen braucht eine GestalterIn mehr technisches und formales Verständnis:

Anfangs muss sie die mathematisch geometrische Beschaffenheit eines Rechtecks bedenken (Abstraktion), „*a closed planar quadrilateral with opposite sides of equal lengths a and b*“ (Weisstein 2008), daraufhin ein generatives System anfertigen (programmieren), welches dieses Objekt auf das Ausgabemedium seiner Wahl zeichnen kann (Formalisierung). Im Anschluss wird der Prozess der Ausführung gestartet (Mechanisierung).

Bis hierhin sind keine Vorteile dieser Methode erkennbar – jedoch existieren entscheidende Faktoren, die ein großes Anwendungsgebiet für generative Gestaltung aufspannen:

1. Die Parametereingabe.

Während bei WYSIWYG Methoden die Parameter mit der Maus festgelegt werden, erlaubt generative Gestaltung² direkte eine Änderung der Parameter. Dies kann mit der Maus passieren, darüberhinaus besteht die Möglichkeit zur Verknüpfung mit anderen Algorithmen und Programmen. Dies ermöglicht unterschiedlichste Eingabegeräte hinzuzuziehen, und Höhe und Breite des Rechtecks zum Beispiel mit einer Telefonwählscheibe zu steuern, oder aus einer Datenbank oder einem Audiosignal herauszulesen.

2. Die Wiederholbarkeit.

Es ist ohne weiteres möglich mit Hilfe des gleichen Algorithmus 100 Rechtecke zu erzeugen. Diese können mit horizontal, vertikal oder zufällig angeordnet werden, ohne dass jedes Rechteck einzeln positioniert werden muss.

WYSIWYG Programme und generative Methoden können unter Umständen gleiche Bilder erzeugen – jedoch macht es für die Umsetzung einer Idee durchaus Sinn darüber nachzudenken, welcher der beiden Methoden man sich bedient. Beiden Verfahren können Anwendungsgebiete zugeordnet werden, die exklusiv oder leichter umzusetzen sind, als mit der jeweils konträren Methode.

Im weiteren Verlauf wird historisch hergeleitet, wie sich diese Verfahren entwickelt haben.

² Zumindest in vielen aktuellen Programmierumgebungen (VVVV, Processing)

1.2 Entstehung

Im folgenden Abschnitt wird untersucht, woraus sich generative Gestaltungsverfahren historisch ableiten lassen. Dabei wird eine Kategorisierung vorgenommen, die formale Vorreiter, kulturelle Voraussetzungen und technische Entwicklungen getrennt betrachtet. Die Grenzen sind dabei fließend, jedoch ist die gedankliche Blockbildung aufgrund rasender Entwicklung in einem kurzen Zeitraum hilfreich.

Betrachtet wird der Zeitraum des 20. und 21. Jahrhunderts – längerfristige kunstgeschichtliche Zusammenhänge können mit Sicherheit gespannt werden, sind für den Verlauf dieser Arbeit jedoch nicht notwendig. Einen ausführlicheren, geschichtlichen Überblick gibt Christian Male in seiner Diplomarbeit „Methodik der apparativen Kunst“, siehe *Male 2005*.

1.2.1 Frühe formale Vorreiter: Konstruktivismus, DeStijl und Bauhaus

Der Begriff des Konstruktivismus fiel erstmals Anfang des 20. Jahrhunderts und bezeichnet die stark auf Abstraktheit und Formen fokussierten Werke, die aus dem Suprematismus hervorgingen. Die Verherrlichung der Technik, und das Streben nach der vollendeten Harmonie spielten inhaltlich eine große Rolle (vgl *BeyArs 2008a*).

Die Methoden der Kreierenden waren berechnend und geplant, die Objektpalette auf Grundformen beschränkt. Zwei Merkmale, die im heutigen generativen Design wiederzufinden sind.

Die Ikone des Konstruktivismus schuf Kasimir Malewitsch mit dem schwarzen Quadrat, das den höchsten denkbaren Grad an Reduktion darstellt.

Im gleichen Zeitraum (1917) erschien in Holland ein Magazin mit dem Namen *DeStijl*. Der Name wurde später zum Begriff für die Kunstrichtung, die die Macher des Magazins prägten. Die formalen Kriterien wurden überaus dogmatisch kommuniziert – vor allem Piet Mondrian hatte eine sehr gefestigte Vorstellungen einer Klassifizierung: „*Bildnerische Mittel sind die Fläche oder – in der Architektur – das rechtwinkelige Prisma in den Primärfarben (Rot, Blau, Gelb) und den so genannten Nicht-Farben (Weiß, Schwarz, Grau)*“ (*BeyArs 2008b*). Zusammen mit Theo van Doesburg und einer sehr wechselnden Anhängerschaft, veröffentlichten sie das

Kunstjournal ca. 10 Jahre lang jeden Monat, um ihre Form- und Flächenforschungen einer Öffentlichkeit zu präsentieren.

Das 1919 gegründete Bauhaus widmete sich der Zusammenführung von freier und angewandter Kunst, von Theorie und Praxis. „*Die Schule soll allmählich in der Werkstatt aufgehen*“ (Gropius 1919), meint das Manifest von Gründer Walter Gropius. Dies hatte zur Folge dass die Grenze zwischen beruflichen Klassen in den Lehrräumen des Bauhaus aufgelöst schien, und dort Akademiker und Handwerker aufeinander trafen. Die stilistisch reduzierte Formensprache der Malerei ging im Bauhaus größtenteils aus den genannten vorhergehenden Strömungen hervor, die Disziplinen waren jedoch vielfältiger: Architektur, Plastik und Typographie spielten zusätzlich eine wichtige Rolle.

Die drei hier aufgezählten Strömungen, die allesamt im ersten Drittel des 20. Jahrhunderts anzusiedeln sind, haben die Gemeinsamkeit die formalen Bestandteile der Kunst zu analysieren. Dies ist eine Gemeinsamkeit zur generativen Gestaltung, da diese in den Schritten der Abstraktion und Formalisierung vergleichbare Analysen verlangt. Zusätzlich lässt sich erkennen, dass bei generativer Gestaltung, wie bei den hier aufgezählten Strömungen, Kombinationen von Grundformen (so genannten Primitiven) die visuelle Ausdrucksform prägen. Von einer direkten Vorstufe generativer Gestaltung zu sprechen wäre übertrieben – die hier genannten Strömungen haben jedoch einen entscheidenden Teil dazu beigetragen, Betrachter an formal reduzierte Bilder heranzuführen.

1.2.2 **Kulturell-künsterische Auseinandersetzungen mit Apparaten.**

„*Allgemein gehen wir davon aus, dass Künstler Kunst machen – was aber geschieht, wenn Maschinen Kunst produzieren? Werden aus Künstlern dann Ingenieure?*“

(Schirn Kunsthalle 2007a, 1)

Den Schöpfungsprozess von Kunstwerken einer Maschine zu übertragen ist ein, kunstgeschichtlich betrachtet, radikaler Schritt, da die Idee des Originalkunstwerkes dadurch in die Optionalität abrutscht (vgl Schirn

Kunsthalle 2007a, 1). Die technischen Möglichkeiten primitiver Zeichenmaschinen waren spätestens seit der Industrialisierung vorhanden – das notwendige Streben diese Tabus der Kunst zu brechen kam allerdings erst nach dem Zweiten Weltkrieg auf (vgl Schirn Kunsthalle 2007a, 1).

1.2.2.1 Autorenschaft in der Maschinenkunst

Mitbegründer der Maschinenkunst ist Jean Tinguely, dessen Zeichenmaschinen, die Méta-Matics Ende der 50er Jahre Aufsehen erregten. Mit Betitelungen wie Méta-Malevitch und Méta-Kandinsky bezog er direkt Stellung gegenüber seinen Vorfahren, und betitelt seine Maschinen (und nicht sich selbst) als deren Nachfolger (meta = gr. nach).

Die Frage nach der Autorenschaft von Kunstwerken ist somit kein hineininterpretiertes Beiwerk, sondern wurde von den Künstlern selbst gestellt.

„Was bedeutet der scheinbare Rückzug des Künstlers aus dem kreativen Akt, und welche Konsequenzen resultieren daraus für Originalität und Einzigartigkeit des Kunstwerks?“

(Schirn Kunsthalle 2007a, 1)

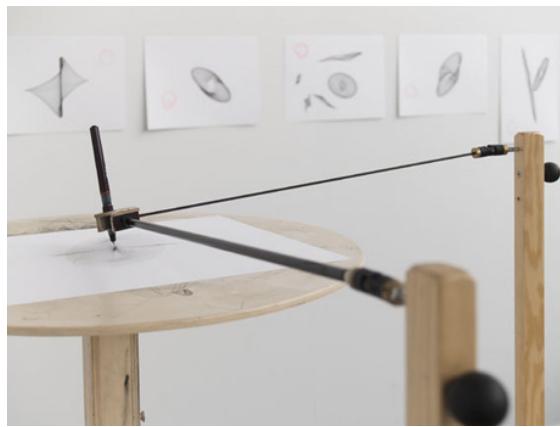
Man würde aus heutiger Sicht nicht behaupten, dass sich Maschinen dieser rudimentären, mechanischen Bauart auf eine Weise verselbstständigen können, die sie selbst zum kreativen Schöpfer der so entstehenden Bilder macht. Vielmehr führen sie die Ideen ihres menschlichen Erschaffers aus – auf Bedarf entweder exakt reproduzierbar, oder mit veränderten Ausgangswerten, die in das vom Mensch erstellte Regelwerk eingespeist werden.

„Nie wird es dem Künstler (...) gelingen, endgültig aus dem Werk zu verschwinden. Die Kunst produzierende Maschine bleibt ein Werkzeug, solange sie sich in den Parametern des Künstlers bewegt. Erst in dem Moment, in dem sie eigenständig handelt und auf Situationen autark reagiert, kann sich die Frage nach der Autorschaft ändern.“

(Schirn Kunsthalle 2007a, 3)

1.2.2.2 Der Parameterbegriff in der Maschinenkunst

Ein wichtiger Aspekt der Maschinenkunst, betrachtet in Hinsicht auf generative Gestaltungsstrategien, ist die Reproduzierbarkeit mit Hilfe von gleichen oder verschiedenen Parametern. Dies wird deutlicher wenn man ein konkretes Beispiel zur Betrachtung hinzuzieht:



img 1.2.01



img 1.2.02

Die Arbeit „The endless study“ von Olafur Eliasson (siehe *img 1.2.01* und *img 1.2.02*) besteht aus einer Art beweglichem Tisch, auf den ein Blatt Papier gespannt wird. Darüber ist eine Halterung für einen Stift angebracht. Der Tisch kann frei schwingen, dh wenn er aus seinem ausbalancierten Ruhezustand in eine beliebige Position gebracht wird, schwingt er wieder in den Ruhezustand zurück. Diese Schwingung ist nicht nur auf einer Achse möglich, sondern kreisförmig. Dabei setzt der Stift, je nach Position des Tisches, entweder auf das Papier auf, oder nicht.

Das Regelwerk des Apparats wird hier definiert durch die Art der Aufhängung und die darauf wirkende Physik. Als Startparameter stehen dem Benutzer verschiedene Faktoren zur Verfügung: die Position des Blattes auf dem Tisch, die Entfernung des Tisches aus seinem Ruhepunkt in den beiden Achsen, die Startgeschwindigkeit beim Loslassen, sowie die Höhe des eingespannten Stiftes über dem Papier.

Sind diese Parameter bei mehrfachem Durchlauf gleich, entsteht theoretisch immer das gleiche Bild. Sobald es zu einer Änderung an mindestens einem dieser Ausgangswerte kommt, wird das Bild anders aussehen. In diesem Fall ist die exakte Reproduzierbarkeit in der Praxis schwer zu erreichen – das Konzept, das auch in der generativen Gestaltung zu finden ist, wird aber sehr deutlich.

1.2.2.3 Kunst trifft Wissenschaft: OpArt

Anfang der 60er Jahre entstand die OpArt, die sich mit menschlicher Wahrnehmung und deren Überschreitung durch optische Phänomene beschäftigte. Der Aspekt der Annäherung von Wissenschaft und Kunst spielte dabei eine große Rolle: Wissenschaftliche Erkenntnisse, in dem Fall über die menschliche Wahrnehmung wurden als Basis für künstlerische Arbeiten benutzt (vgl Schirn Kunsthalle 2007b, 1).

Die Grenzen der Wahrnehmung wurden von der Wissenschaft erforscht, und die Kunst verwendete sie als Experimentierfeld.

Diese Annäherung passierte auch umgekehrt: Die Wissenschaft formulierte die Grenzen, wobei Arbeiten mit künstlerischem Ausdruck entstanden.

Dieselbe Art der beidseitigen Annäherung der Begriffe Kunst und Wissenschaft wird im Abschnitt über die Computergrafik erneut auftreten und spielt auch weiterhin, bis in die zeitgenössische Kunst, eine bedeutende Rolle.

1.2.3 Digitalisierung

1.2.3.1 Anfänge der Computergrafik

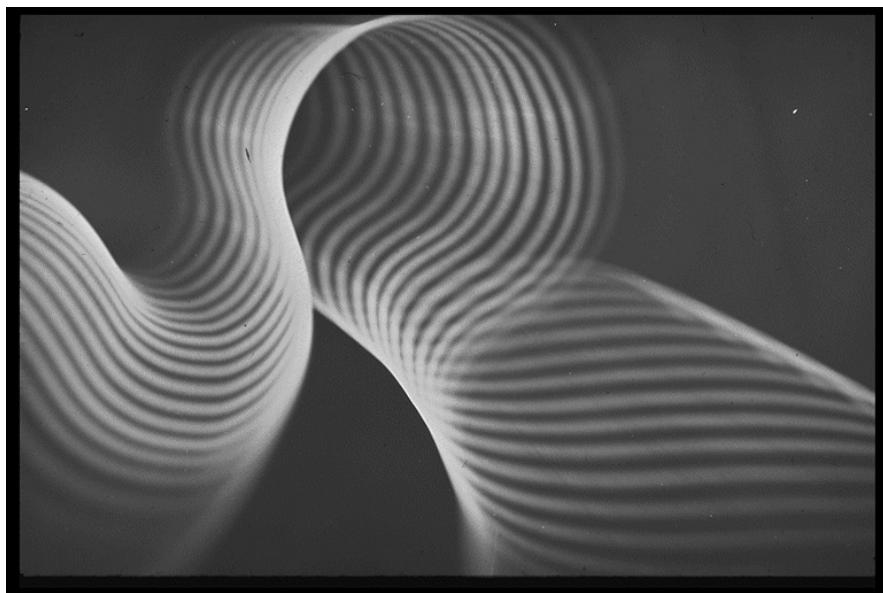
Die Anfänge der Computergrafik stehen in einem sehr direkten Zusammenhang mit der generativen Gestaltung von heute. Es folgt ein Abriss dieser Entwicklung.

Beim Fixieren eines Startpunkts der Computergrafik stellt sich die Frage, ab wann man überhaupt von einer Grafik sprechen kann. Die analoge Herangehensweise der Maschinenkunst hatte klar das Ziel, Maschinen künstlerisch arbeiten zu lassen. Die digitale Herangehensweise näherte sich anfangs sehr von der technischen Seite an die Erstellung von visuellen Produkten. Dies ist darauf zurückzuführen, dass die benötigten Geräte anfangs nur Industrie und Forschung zur Verfügung standen und deswegen nur schwer künstlerische Relevanz darin gesehen wurde.

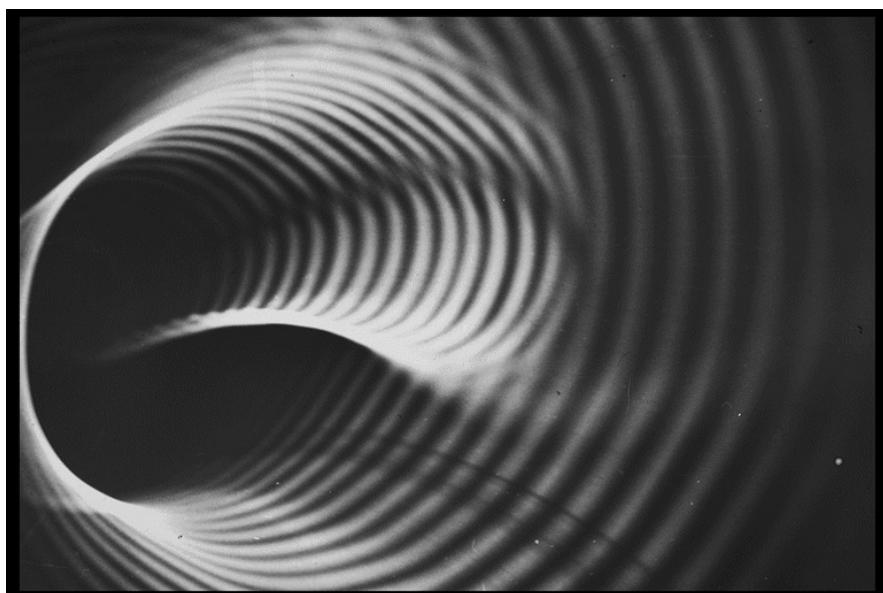
Um von einer Grafik zu sprechen müssen also „*Figuren, (...) aus dem wissenschaftlich-technischen Zusammenhang gelöst und als ästhetische Gebilde wahrgenommen werden*“ (Klütsch 2007, 230).

Vorreiter, Visionär und soziale Schnittstelle der frühen Computergrafik war der Wiener Physiker Herbert W. Franke. Neben seinen naturwissenschaftlichen Neigungen zeigte sich bereits im Studium sein weit gefächertes Interesse – er studierte Physik, Chemie, Psychologie und Philosophie. Während seiner technischen Tätigkeit als Ingenieur war er bereits stets daran interessiert, ästhetische Bilder zu erzeugen.

Herbert W. Franke ist deshalb ein geeignetes Beispiel, in die Computergrafik einzuleiten, da er das Gebiet mit analogen Experimenten (siehe zB *img 1.2.03*) eigens herleitet. Seine frühen fotografischen Experimente beschäftigten sich mit abstrakten Bildern, die aufgrund von technischen Vorgängen entstanden sind. Die Serie *Lichtformen* (siehe *img 1.2.03* und *img 1.2.04*) erstellte Franke mit Hilfe von schwingenden und vibrierenden Drähten, die in abgedunkelten Räumen zum Glühen gebracht wurden. Stets auf dem Medium Fotografie festgehalten, und dessen Eigenschaften wie Langzeitbelichtung und Bewegungsunschärfe zu Nutze machend, kreierte er Bilder, die an Schwingungs- und Wellenbilder erinnern (vgl Piehler 2007, 64ff). Ähnliche visuelle Phänomene treten in der heutigen Computergrafik auf. Diese fotografischen Experimente können aufgrund der Reproduzierbarkeit des Prozesses heute als Vorbereitung zu den elektronischen Grafiken betrachtet werden (vgl Franke 2007, 108).



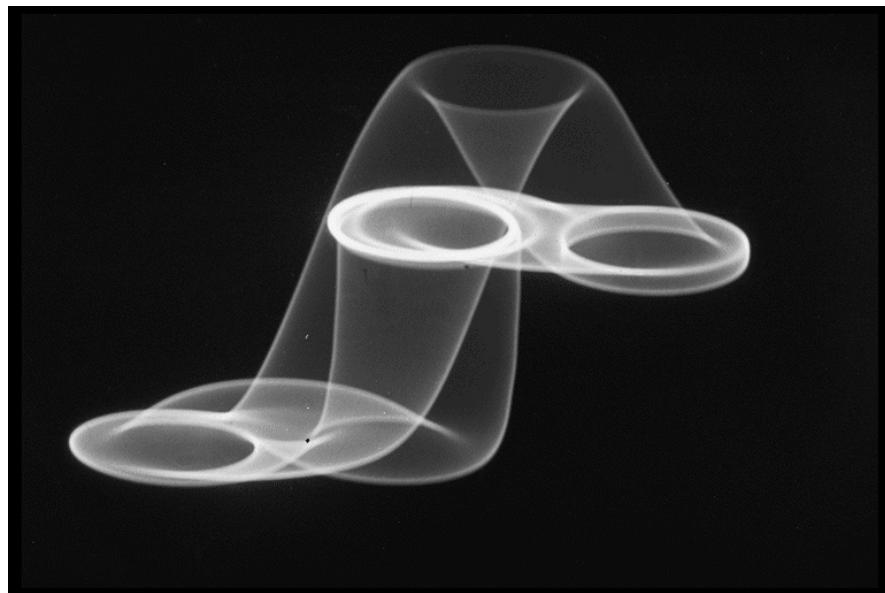
img 1.2.03



img 1.2.04

Schwingungen sollten auch im weiteren Verlauf eine Rolle spielen: Im Jahre 1955 verwendete Franke einen Oszillographen, um damit Lissajous Figuren auf dessen Mattscheibe zu werfen. „*Lissajous-Figuren sind Kurvengraphen, die durch Überlagerung harmonischer Schwingungen entstehen*“ (Wikipedia 2008b). Bei konstanter Veränderung der Phase kann dabei ein dreidimensionaler Eindruck entstehen. Im Unterschied zu anderen durchgeföhrten Experimenten mit Oszillographen, die durchaus bereits mit dem Ziel der Bilderstellung fotografisch festgehalten wurden (zB vom amerikanischen Mathematiker und Künstler Ben F. Laposky 1952)

benutzte Franke für künstlerische Zwecke modifizierte Technik: die Schwingungen wurden bei Frankes Apparat nicht gemessen, sondern mit eigens entwickelten Steuermechanismen erzeugt. So konnte in Echtzeit auf ein möglichst ästhetisches Ergebnis hingearbeitet werden (vgl Piehler 2007, 64ff). Der Diskurs der technischen Versuchsanordnung wurde somit verlassen, und die Gestaltung trat in den Vordergrund. „*Bei Franke und Laposky sind Lissajous-Figuren nicht mehr Indikatoren von Phasenverschiebungen, sondern Bilder des ästhetischen Genießens*“ (Rosen/Weibel 2007, 188).



img 1.2.05

Herbert W. Franke über den Entstehungsprozess:

„*Es ist ein Gefühl, ähnlich dem was ein improvisierender Musiker empfindet. Man sieht die abenteuerlichsten Formen entstehen und zerfließen, wandern und kreisen, sich zusammenziehen und lösen. Ich habe stundenlang vor der magischen Scheibe gesessen und die hellgrün leuchtenden Linien verfolgt – und darüber vergessen, daß neben mir die Kamera stand, mit der ich diese Eindrücke festhalten wollte.*“

(Franke 1957, 28 zit n Rosen/Weibel 2007, 188)

Diese Bilder sind ausreichend aus dem wissenschaftlich-technischen Zusammenhang gelöst, um Beachtung im künstlerischen Diskurs zu finden, und wenig später in den musealen Raum gebracht zu werden.

Die ersten am Computer erzeugten Grafiken kamen 1959 von dem Japaner Hiroshi Kawano – kurz darauf ward auch erstmals die Bezeichnung dafür gefunden: William A. Fletcher führte, während er an Vorstufen der Entwicklung des Zeichenprogramms CAD beteiligt war, den Begriff *computer graphics* ein (vgl Klütsch 2007, 232).

1962 wurde, ebenfalls am MIT, die erste Zeichenanwendung entwickelt, die dem User ermöglichte, ohne große Programmierkenntnisse digitale Bilder zu erzeugen:

„Das elektronische Werkzeug des Lichtgriffels, das einem Zeichenstift nach-empfunden ist, lenkt durch eine Linse das Licht des Bildschirms, über den es bewegt wird, auf eine fotoelektronische Zelle.“
(Klütsch 2007, 234)

Mit der Entwicklung der Zeichenmaschine „Graphomat Z64“ war es kurz darauf möglich, die digitalen Bildern auf Papier zu bringen, ohne wie bisher den Umweg über die Fotografie nehmen zu müssen.

Die kontinuierlich wachsenden Möglichkeiten der digitalen Bildgestaltung weckten mehr und mehr Interesse bei Künstlern.

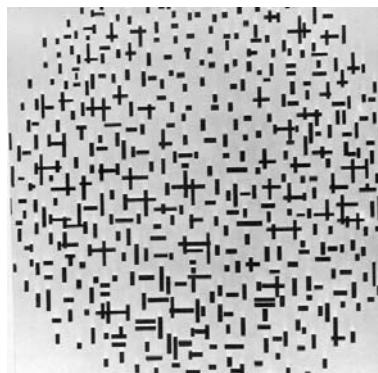
Interessanterweise wurde gerade in den Anfangsjahren ein Trend verfolgt, der bereits in der Maschinenkunst aufgefallen ist: es wurde versucht bekannte Kunstwerke mit den neuen Mitteln nachzuahmen:

„Von Mondrian (...) ist es nur ein Schritt zum Computer“ (Nees 1969, 7), bewies A. Michael Noll mit seinen Computer Compositions with Lines (img 1.2.06) – dabei handelt es sich um digitale Pendants von Piet Mondrians Komposition mit Linien (img 1.2.07) von 1917. Dass Noll den Plural bei seinem Titel verwendete (Compositions) ist kein Versehen, sondern wichtiger inhaltlicher und formaler Bestandteil: er beschrieb das Bildprogramm als das Werk, und nicht einen visuellen Repräsentanten davon. *„An die Stelle des Bildes trat die Bildklasse.“* (Nierhoff-Wielk 2007, 23).

Die Vielfalt an visuellen Repräsentanten wurde durch den Parameter Zufall ermöglicht, der auf die Eigenschaften von Mondrians Gemälde angewendet wurde: Länge und Ausrichtung der Linien konnten zufällig variieren und verschiedene Versionen der Komposition erzeugen. Bei dieser Arbeit wird deutlich, dass hier die Prinzipien der mechanistischen Weltauffassung in den Gestaltungsprozess transferiert wurden: Abstraktion, Formalisierung, Mechanisierung.



img 1.2.06



img 1.2.07

„Dem traditionellen Künstler gehe es um die eine Zeichnung. Der Programmierer beschreibt das Schema aller Zeichnungen.“

(Nierhoff-Wielk 2007, 23)

Der Programmierer muss multiple Kompetenzen aufbringen, um einen solchen Gestaltungsprozess alleine zurücklegen zu können.

Die Öffentlichkeit und der damalige Zeitgeist brachten dieses Verständnis noch nicht auf. Die Maschinen wurden als das schöpferische Instrument angesehen, das zwar von Menschen überwacht (im Sinne von an- und abgeschaltet), aber nicht vom Menschen bedient wird (wie es zum Beispiel bei Musikinstrumenten der Fall ist). Ein Artikel der Zeitschrift SPIEGEL aus dem Jahre 1965 mit dem Titel „Bald krumme Linien“, illustriert dies auf eine aus heutiger Sicht charmante und überaus ehrliche Art:

„Der kombinierte Rechen- und Zeichenautomat ist eine jener neuen pseudoschöpferischen Maschinen, die dem Menschen nicht nur beim Subtrahieren, Multiplizieren, Wurzelziehen und Differentialrechnen helfen, sondern bereits selber Lyrik schreiben, Romane verfassen, Sonaten komponieren, aus dem Chinesischen übersetzen, Schach spielen und ihren wissenschaftlichen Herren als Gesprächspartner dienen.“

Der künstliche Zeichenkünstler (...) steht in einem Zimmer des Recheninstituts der Technischen Hochschule Stuttgart und wird dort von einem wissenschaftlichen Assistenten, dem Mathematiker Frieder Nake, 26, beaufsichtigt“

(Der Spiegel 1965, 151)

Der Mensch wurde als Assistent, und die Maschine als Zeichenkünstler dargestellt. Ähnlich wie Louis Couffignals Vision in *Denkmaschine*, nach der der Mensch das Denken und Handeln nach und nach den Maschinen übergibt, und nur noch benötigt wird „um die [Loch-]Kartenpakete von einer Maschine zur anderen zu bringen, und sich seine Anstrengung darauf reduziert, zu verhindern dass die Karten innerhalb eines Paketes durcheinander kommen“ (Couffignal 1955, 33 zit n Rosen/Weibel 2007, 182).

Kein Wunder also, dass künstlerische und kreative Einflüsse dem Menschen in diesem Zusammenhang abgesprochen wurden.

Nichtsdestotrotz vereinte die aufkommende, mit Art-and-Technology betitelte Tendenz, Ingenieure, Künstler und Maschinen mehr und mehr auch im musealen Raum. Im weiteren Verlauf der 60er Jahre bestätigte sich der Trend durch immer populärer werdende Veranstaltungen, der 1968 seinen vorläufigen Höhepunkt mit bedeutenden Ausstellungen im MoMA/New York und der Cybernetic Serendipity in London findet (vgl Klütsch 2007, 250-254).

Aufgrund dem Einzug in die Museen ist in geschichtlichen Abhandlung häufig der Begriff der frühen Computerkunst statt dem der Computergrafik zu finden.

Die kontinuierliche technologische Entwicklung öffnete das Feld der Informationstechnologie mehr und mehr für Jedermann.

Ende der 70er Jahre wurden Computer für Firmen erschwinglicher, was von der Pionierarbeit der frühen Computergrafik, die nachts und an Wochenenden in großen Rechenzentren der Universitäten und Großkonzernen stattfand, hin zu heimischem Arbeiten, führte. Dieser Wandel hatte zur Folge, dass im Laufe der Zeit nicht nur Menschen mit Beziehungen zu Industrie und Hochschulen Computergrafik erstellen konnten.

Mit Gründung der Ars Electronica im Jahre 1979, ein Festival, das sich auf der Grenze zwischen Kunst, Technologie und Gesellschaft ansiedelt, bekam das Genre eine Plattform, die konstanter Begleiter der Bewegung wurde. Im Laufe der Zeit ist das Konzept neben dem jährlichen Festival um eine Preisverleihung, ein Museum und ein Labor erweitert worden. Einer der Ars Electronica Gründer ist der Computergrafik Pionier Herbert W. Franke.

1.2.3.2 Der Weg zur aktuellen Computergrafik

In den 1980er Jahren wurden aufgrund der technischen Errungenschaft der graphischen Benutzeroberfläche neue Konzepte in der Erstellung von Grafik verfolgt. „What you see is what you get“, kurz WYSIWYG, war der Begriff, dem sich Softwareentwickler in dieser Zeit widmeten. Die dafür notwendige Computermaus wurde zwar bereits 1968 vorgestellt, feierte ihren Durchbruch jedoch erst mit dem Erscheinen von Apples Macintosh im Jahre 1984. Die Entwickler von Softwarepaketen wie Adobes Photoshop, Illustrator, und Corel Painter haben mit Hilfe neuer Formate und Eingabegeräte im Laufe der Zeit Möglichkeiten eröffnet, die Grafiken bis hin zum Fotorealismus hervorbrachten.

Grafik direkt zu Programmieren war in diesem Zeitraum eher eine Randerscheinung – im Fokus der Entwickler stand in erster Linie das Kreieren von Userinterfaces.

Das MIT war Mitte der 90er Jahren die Einrichtung, die generative Gestaltungsstrategien verfolgte und wieder ins öffentliche Interesse rückte. Publikationen des dort lehrenden John Maeda wie „Design by Numbers“ (siehe Maeda 1999) und „Creative Code“ (siehe Maeda 2004) vereinen generative Gestaltungsstrategien der Computerkunst mit den Technologien der Gegenwart. Am MIT entstand auch die prominente Verbindung Maedas zu den beiden Studenten Casey Reas und Ben Fry. Reas und Fry haben mit ihrer dort eigens entwickelten Programmierumgebung *Processing* die generative Gestaltungsszene nachhaltig geprägt.

Tiefergehende Informationen zu aktueller Software, die das Erzeugen generativer Arbeiten ermöglicht, werden im Kapitel über die Werkzeuge (1.3) aufgezeigt.

Die Entstehung von generativer Gestaltung ist, wie nun sichtbar wurde, ein langer Prozess. Interessanterweise ist festzustellen, dass die Anfänge der Computergrafik mit der heute im Trend liegenden generativen Herangehensweise an Gestaltung mehr gemeinsam haben, als die Entwicklungen im Zusammenhang mit dem WYSIWYG Konzept der letzten drei Jahrzente. Eine Rückkehr zu den algorithmischen Wurzeln ist mit neuen, hochtechnisierten Mitteln in vollem Gange und wird mehr und mehr zu einer festen Instanz im Designdiskurs.

1.3 Werkzeuge

Nachdem im bisherigen Verlauf deutlich geworden ist, was generative Gestaltung bedeutet, und wie sich der Prozess des Gestaltens dabei im Vergleich zu herkömmlichen Methoden verhält, stellt sich die Frage, welche Möglichkeiten GestalterInnen heute zur Verfügung stehen, generative Gestaltung hervorzubringen.

Generatives Gestalten setzt nicht automatisch voraus, dass GestalterInnen Programme von Grund auf erstellen, um Algorithmen einzuarbeiten. Man unterscheidet zwischen zwei Verfahren:

- vorhandene Software mit Scripts, Plugins oder hacking-ähnlichen Eingriffen um generative Schnittstellen zu erweitern,
oder
- eigene Programme, mit Hilfe einer Programmierumgebung von Grund auf zu entwickeln.

Die gesamte Kette der Informationsverarbeitung heutiger Computer bietet Punkte und Schnittstellen, die von programmierenden GestalterInnen genutzt werden können, um externe und eigene Parameter einzuarbeiten, die das visuelle Ergebnis verändern.

Auch herkömmliche WYSIWYG Programme, wie zum Beispiel Adobe Illustrator, bieten die Möglichkeit eigene Plugins oder Scripts einzubinden. Unter Plugins und Scripts versteht man kleine Programme, die es ermöglichen, eigene Algorithmen in vorhandene Software zu implementieren.

Im Folgenden werden beide Prinzipien anhand von Beispielen erläutert.

1.3.1 Scriptographer

Als Beispiel, wie man Schnittstellen von WYSIWYG Software generativ nutzen kann, wird Adobe Illustrator mit dem Plugin Scriptographer¹ betrachtet.

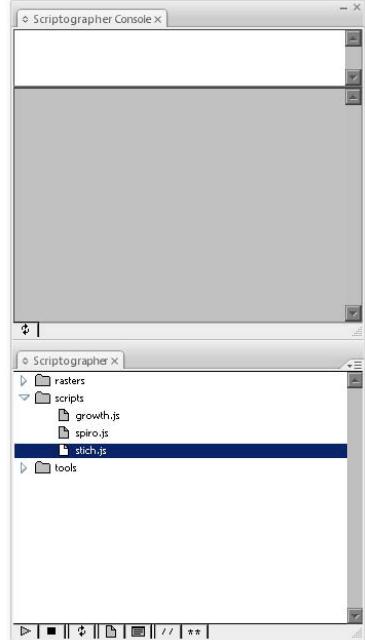
Illustrator ist ein gängiges WYSIWYG Tool zum Erstellen und Bearbeiten von Vektorgrafik. Das Prinzip der Vektorgrafik ist, Objekte nicht

¹ Frei erhältlich unter: <http://www.scriptographer.com/>

auf Pixelraster basierend zu erzeugen und bearbeiten, sondern aufgrund von mathematischen Operationen (Kurven und Formen) zu beschreiben. Dies ermöglicht uneingeschränkte Skalierbarkeit, und macht es zum Standard für Logo- und Zeichenentwicklung. Die Bedienung erfolgt im gängigen WYSIWYG Stil: will die BenutzerIn ein Rechteck zeichnen, wählt sie mit der Maus das Werkzeug aus, und zieht, entsprechend nach ihren Wünschen (Höhe, Breite, Position), ein Rechteck auf dem Bildschirm auf.

Das Scriptographer Plugin erweitert die Funktionen von Illustrator um eine JavaScript Schnittstelle. Das heißt, in der Benutzeroberfläche können zwei zusätzliche Fenster angezeigt werden: eines davon erlaubt, vorgefertigte Algorithmen aufzurufen und das andere bietet die Möglichkeit eigene Algorithmen in Form von JavaScript Code einzugeben.

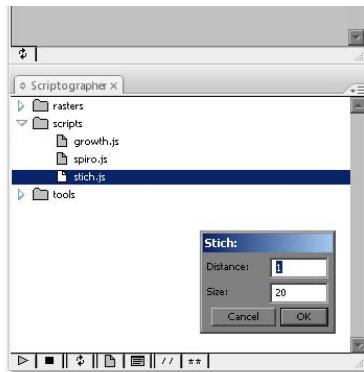
JavaScript ist eine objektorientierte Programmiersprache, die primär bei Webanwendungen eingesetzt wird. Ein genaues Verständnis davon ist an dieser Stelle nicht erforderlich, da sich das folgende Beispiel auf die Anwendung von vorgefertigten Algorithmen beschränkt.



img 1.2.01 zeigt die Eingabekonsole für eigene Algorithmen, sowie die Auswahlmöglichkeit von vorgefertigten Algorithmen, die sich als .js Datei im Plugin Ordner befinden. Klickt man einen Eintrag daraus doppelt an, öffnet sich der Programmcode in einem Texteditor. Ein Klick auf den Play Button führt die Funktion aus. In vielen Fällen handelt es sich um Operationen, die auf bereits vorhandene Objekte angewendet werden.

Für folgendes Beispiel wird die Funktion `stich.js` auf ein im Illustrator gezeichnetes Quadrat angewendet.

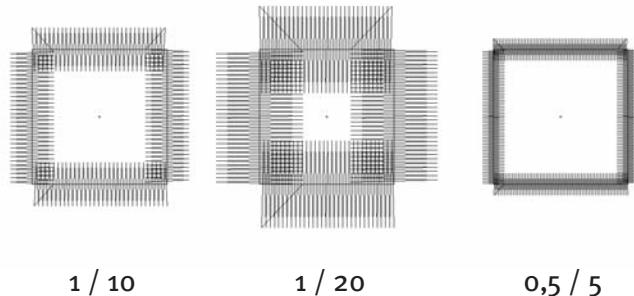
Wenn die Funktion bei markiertem Quadrat ausgeführt wird, erscheint zunächst ein Eingabefeld für die Parameter. Diese sind im Falle des Algorithmus `stich.js` *Distanz* und *Breite* (siehe *img 1.3.02*).



Mit einem Doppelklick auf den Eintrag, sieht man den Code welcher hinter der Funktion steckt:

```
var sel = document.getMatchingItems(Path, { selected: true });
if (sel.length > 0) {
    values = Dialog.prompt("Stich:", [
        { value: 1, description: "Distance", width: 50 },
        { value: 10, description: "Size", width: 50 }
    ]);
    if (values) {
        var dist = values[0];
        var size = values[1];
        for (var j = 0; j < sel.length; j++) {
            var art = sel[j];
            art = art.clone();
            art.curvesToPoints(dist, 10000);
            var mul = 1;
            var res = new Path();
            for (var i = 0, j = art.curves.length; i < j; i++) {
                var curve = art.curves[i];
                var pt = curve.getPoint(0);
                var n = curve.getNormal(0);
                if (n.x != 0 || n.y != 0) {
                    n = n.normalize(size);
                    res.segments.add(pt.add(n.multiply(mul)));
                    mul *= -1;
                }
            }
            art.remove();
        }
    } else {
        Dialog.alert("Please select a path.");
    }
}
```

Folgende Abbildung zeigt die Resultate von drei verschiedenen Durchläufen, mit jeweils verschiedenen Parametern:



img 1.3.03

Dieses Beispiel verdeutlicht, wie man in Illustrator mit JavaScript parameterbasierte, generative Eingriffe auf Vektorgrafik Elemente vornehmen kann. Allerdings bewegt man sich als GestalterIn weiterhin innerhalb der ausgereizten Grenzen von Illustrator, und kann das Feld der Vektorgrafik nicht verlassen.

1.3.2 **VVVV**

„vvvv is a toolkit for real time video synthesis. It is designed to facilitate the handling of large media environments with physical interfaces, real-time motion graphics, audio and video that can interact with many users simultaneously.“

(vvvv.org 2008a)

VVVV ist eine visuelle Programmiersprache, das heißt Programme werden mit Hilfe einer grafischen Benutzeroberfläche erstellt, und nicht durch das Schreiben von Programmcode. Mehr zu diesem Programmierkonzept in VVVV im *Kapitel 1.3.2.1*.

Die Entstehung von VVVV begann 1998 im Frankfurter Designbüro MESO. Ursprünglich als firmeninterner Toolkit geplant, war VVVV Anfangs eine codebasierte Sammlung von Programm-Fragmenten, die in der Firmenpraxis häufig verwendet wurden. Das Anwendungsgebiet besteht in erster Linie aus Multimediainstalltionen, weswegen der

Entwicklungsfookus von VVVV auf der Implementierung verschiedener Eingabegeräte, sowie einem graphisch hochwertigen Output liegt.

2001 wurde ein graphisches Benutzerinterface eingeführt, und die Software der Öffentlichkeit zugänglich gemacht (vgl VVVV.org 2007).

VVVV ist frei verfügbar für nicht-kommerzielle Anwendung. Das bedeutet eine voll funktionsfähige Version kann im Internet unter vvvv.org heruntergeladen werden.

VVVV befindet sich seit jeher in der Betaphase. Die aktuelle Version, auf die sich diese Arbeit bezieht ist *vvvv_40beta17*, und liegt der DVD bei.

1.3.2.1 Konzept der visuellen Programmierung in VVVV

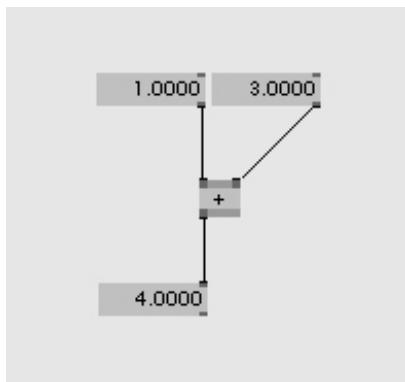
VVVV ist im Feld der visuellen Programmiersprachen anzusiedeln. Konkreter klassifiziert handelt es sich um ein visuelles Programmiersystem mit grafischer Programmiersprache:

„Bei visuellen Programmiersystemen handelt es sich um Entwicklungsumgebungen, welche die Programmierung von Software mit Hilfe einer visuellen Programmiersprache ermöglichen. Diese Systeme sind in der Regel sehr eng mit der verwendeten Sprache verwoben, die Userinterfaces bilden sogar häufig einen Teil der Sprache. Eine Trennung von Werkzeug und Sprache, wie bei klassischen Systemen, ist daher nicht möglich“

(Brüll/Schwarzer 2006, 11)

Begrifflich wird fortan zwischen visueller Programmiersprache, und textbasierter Programmiersprache unterschieden. Nach obiger Definition schließt der Begriff der visuellen Programmiersprache häufig auch die Programmierumgebung (Werkzeug) ein. VVVV ist also gleichzeitig Programmierumgebung, Programmiersprache und Programmiersystem.

Beim visuellen Programmieren in VVVV schreibt man keinen Code, sondern hat eine Benutzeroberfläche zur Verfügung, die das Erstellen von Funktionsblöcken, so genannten Knoten (engl. Nodes) ermöglicht. Knoten führen Funktionen aus. In einfachen Fällen, wie dem Berechnen einer Summe, gehen in den Knoten zwei Werte hinein (Summand 1 und 2), und ein Wert (Summe) kommt heraus (siehe *img 1.3.03*). In VVVV befinden sich die Inputs oben, und die Outputs unten am Knoten. Die Anschlüsse nennt man Pins.



img 1.3.04

Knoten können mit Linien (Datenkanäle) miteinander verbunden werden und somit Werte (oder andere Datentypen) weitergeben.

Dieses, Datenstrom-Diagramm-Prinzip, unterliegt allgemein folgenden Regeln:

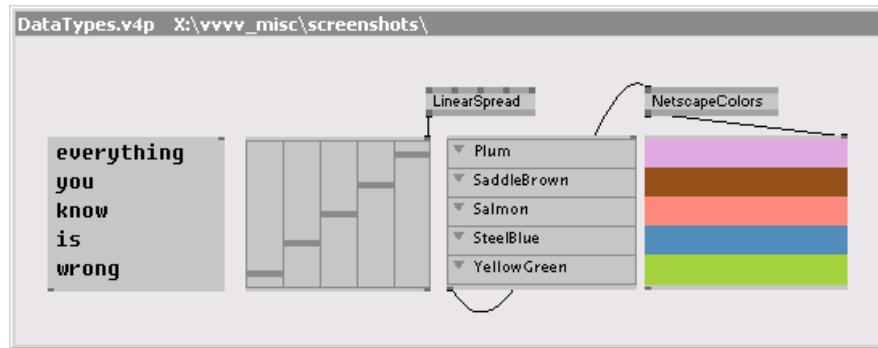
- „- Pro Eingang darf nur ein Datenkanal angeschlossen sein.
- Pro Ausgang können mehrere Datenkanäle angeschlossen werden.
- Datenquellen [Ausgangsparameter] besitzen ausschließlich Ausgänge.
- Datensenken [Ergebnisse] besitzen ausschließlich Eingänge.
- Ein Knoten besitzt mindestens einen Eingang und keinen oder mehrere Ausgänge.
- Jeder Eingang eines Knotens muss mit einem Datenkanal verbunden sein oder eine Datenquelle besitzen. Sonst ist die Operation nicht ausführbar.
- Ein Knoten ohne Ausgänge oder Verbindungen zu anderen Knoten oder Datensenken ist wirkungslos.“

(Heinsch 2007, 16)

Neben VVVV gibt es in diesem Anwendungsgebiet noch weitere, Programmiersprachen, die nach diesem Prinzip funktionieren, zB MAX/MSP und PureData. Die Regeln werden dabei mehr oder weniger streng eingehalten: bei PureData ist es im Gegensatz zu VVVV möglich, an einen Eingang mehrere Datenkanäle anzuschließen.

Datenquellen und Datensenken, sprich Ausgangsparameter und berechnete (Zwischen-)Ergebnisse werden in VVVV in so genannten IO-Boxen (Input-Output Box) eingegeben bzw angezeigt (siehe *img 1.3.04*). Diese IO-Boxen existieren für jeden in VVVV vorhandenen Datentyp: Werte (Value), Unicode Zeichenkombinationen (String), Farben (Color), Auswahllisten (Enumeration) und abstrakte Daten

(Node). Jedem Ein- oder Ausgangspin ist ein fester Datentyp zugeordnet, der nur Verbindungen zum gleichen Typ zulässt (vgl Heinsch 2007, 24).

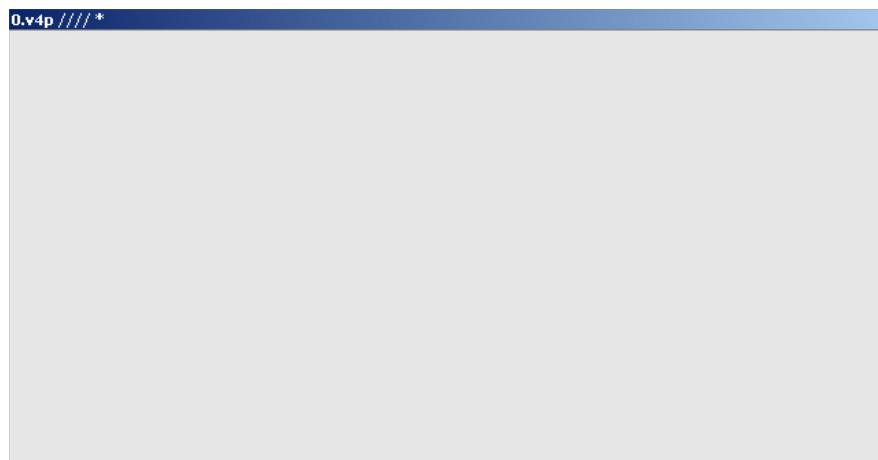


img 1.3.05

Um nicht für jede Werteänderung eine IO-Box anlegen zu müssen, ist es darüberhinaus möglich, über die entsprechenden Input-Pins des Knotens, der eine Werteänderung erfahren soll, mit der rechten Maustaste den Wert direkt am Pin zu ändern.

1.3.2.2 Benutzeroberfläche und Bedienung

Startet man VVVV, so öffnet sich ein graues Fenster ohne Werkzeugeleiste oder andere sichtbare Bedienelemente. Die Interfaceelemente sind versteckt und werden nur auf Anforderung eingeblendet. Die Absicht dahinter ist, den maximal verfügbaren Platz zu nutzen, um auch in großen Applikationen den Überblick zu behalten.



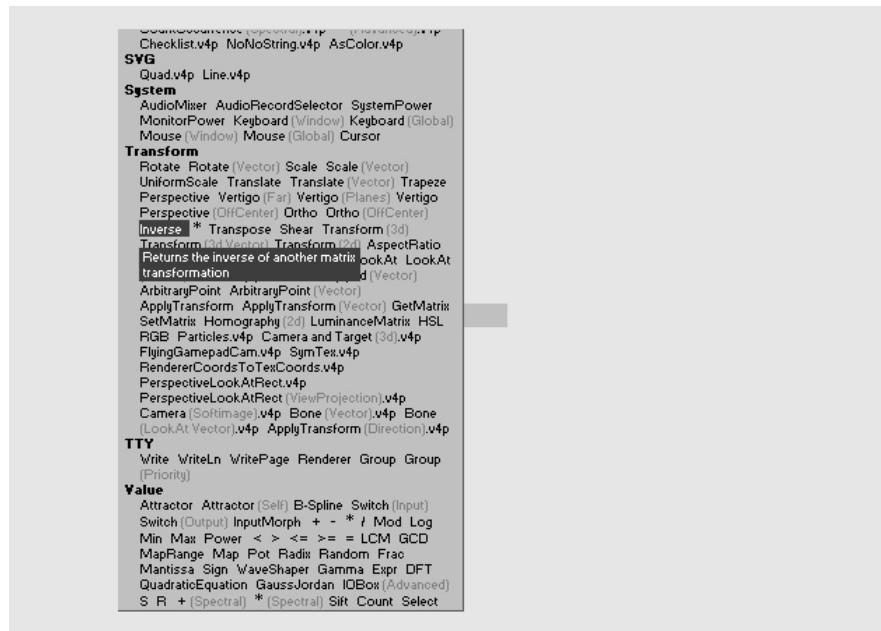
img 1.3.06

Dieses graue Feld (siehe *img 1.3.05*) wird im Arbeitsprozess mit Knoten und Verbindungslien gefüllt. Die so entstehenden Applikationen

nennt man Patch (dt: *Korrektur*), den Prozess dahinter bezeichnet man umgangssprachlich auch als patchen.

Optionen lassen sich über das Hauptmenü (mittlere Maustaste), oder Tastenkürzel aufrufen. Knotenspezifische Einstellungsmöglichkeiten bietet der sogenannte Inspektor, der im weiteren Verlauf oft hinzugezogen wird, um Übersicht über die Funktion von Knoten zu geben..

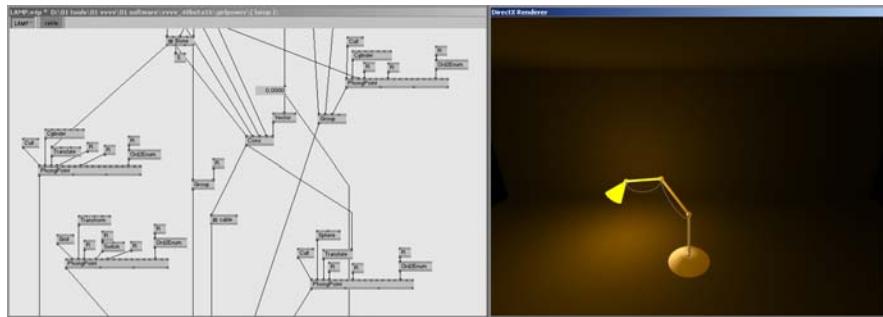
Knoten werden mit Hilfe eines Doppelklicks erstellt. Daraufhin wird ein Eingabefeld geöffnet, in das man den Namen des gewünschten Knotens tippt. Alternativ dazu kann man mit der rechten Maustaste im Eingabefeld eine Knoten-Liste aufrufen (siehe *img 1.3.06*), die entweder alphabetisch oder kategorisch sortiert angezeigt wird.



img 1.3.07

Für grafischen Output wird ein so genannter Renderer benötigt, der die gewünschten Objekte zeichnet. Ein Renderer öffnet sich standardmäßig in einem neuen Fenster, er kann aber je nach Anwendungsfall auch im Patch integriert oder ganz versteckt werden.

Für generative Gestaltung ist einzigst der DirectX Renderer relevant, der sich Microsofts DirectX API (Application Programming Interface, dt. Programmierschnittstelle) bedient, und dadurch komplexen, grafischen Output erzeugen kann. VVVV kann somit die Leistung der im Rechner verbauten Grafikkarte ausschöpfen (vgl *VVVV.org 2008b*).



img 1.3.08

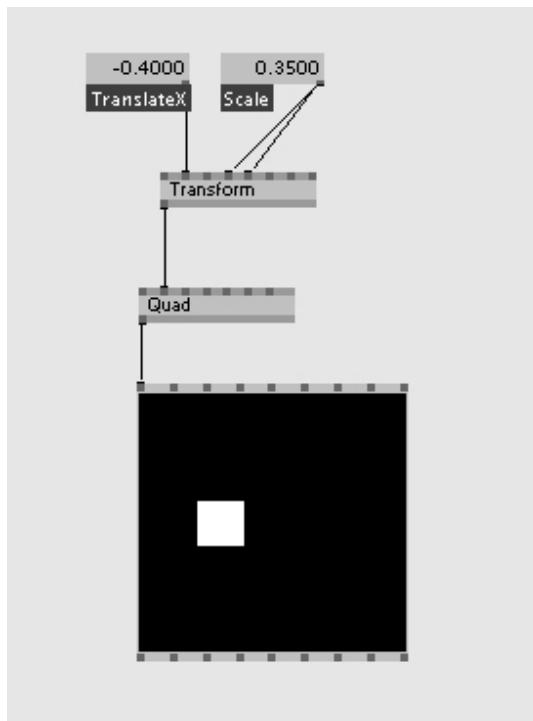
Obige Abbildung (*img 1.3.07*) zeigt einen komplexeren Patch inklusive zugehörigem Renderer (rechts).

VVVV erlaubt es Patches zu verschachteln und so genannte Subpatches zu erstellen. Das ist von Vorteil für Applikationen, die nicht auf den Bildschirm passen. Logisch zusammenhängende Knotenkombinationen können in Subpatches verpackt und wie eigene Knoten behandelt werden. Mit einem Rechtsklick auf den Subpatch, kann jederzeit der Inhalt angezeigt und angepasst werden.

1.3.2.3 generatives Gestalten in VVVV

Gestaltung beginnt in VVVV in den allermeisten Fällen mit geometrischen Grundelementen (Primitives). Diese lassen sich im weiteren Prozess transformieren, vervielfachen oder in ihrer Erscheinung verändern, sowie kombinieren, verfärbten etc.

Das einfachste Beispiel, um die Objekthandhabung in VVVV zu erläutern, ist das Rechteck. Dafür benötigt man in VVVV das „Primitive“ *Quad (DX9)* und einen DirectX Renderer. Wie zusätzliche Transformationen ermöglicht werden, zeigt folgende Abbildung (*img 1.3.08*).

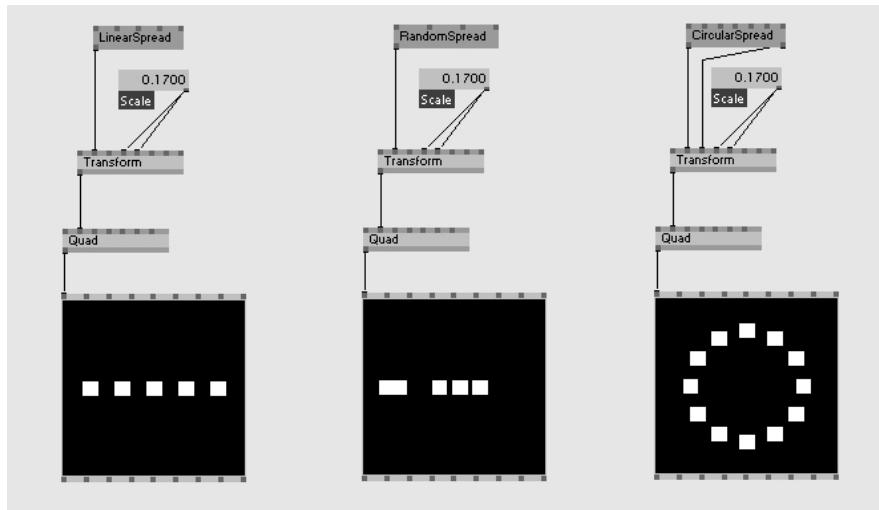


img 1.3.09

Der nächste Schritt zeigt die Handhabung von Wertelisten: die so genannten Spreads. Fast alle Knoten in VVVV besitzen die Fähigkeit, Spreads zu verarbeiten.

Um eine Vielzahl von Werten zu generieren bietet VVVV wertegebende Knoten. Die so erzeugten Listen folgen gewissen Regeln. Das einfachste Beispiel ist der *LinearSpread*, der eindimensionale Werte in einer Spanne linear verteilt (siehe *img 1.3.10* links). Das gleiche Prinzip, mit zufälliger anstatt linearer Verteilung, erfüllt der *RandomSpread* (siehe *img 1.3.10* Mitte).

Der *CircularSpread* (siehe *img 1.3.10* rechts) und der *TypoSpread* geben X und Y Werte aus. Objekte können auf diese Weise in Kreisform oder Buchstabenform angeordnet werden.

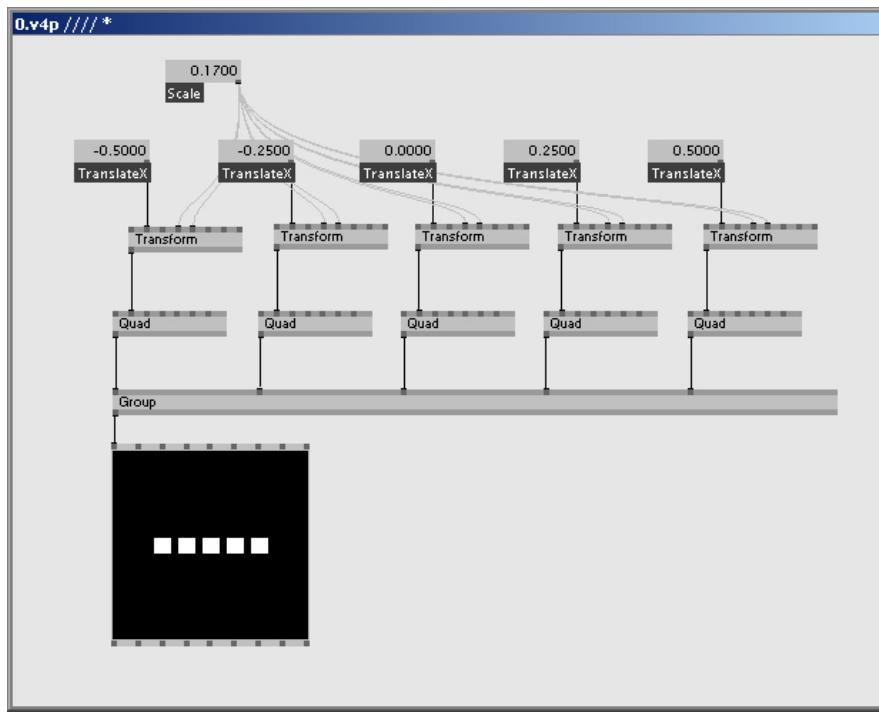


img 1.3.10

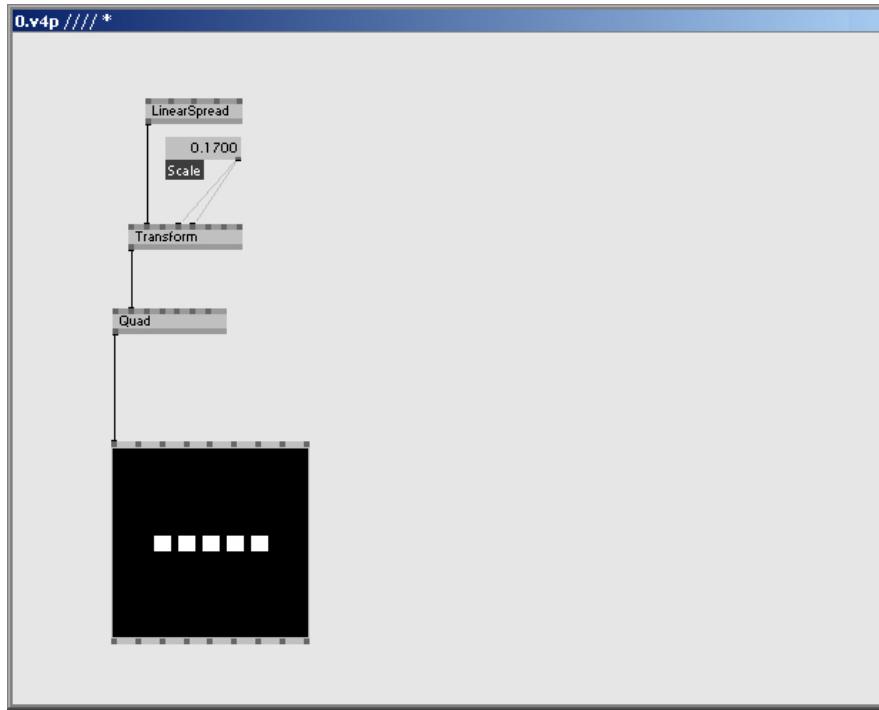
Sofern spezifische Werte benötigt werden, die nicht algorithmisch generiert werden können, können Wertelisten auch in IO-Boxen einge tragen werden.

Sollen also mehrere geometrische Objekte vom gleichen Typ erstellt werden, muss nicht der Knoten *Quad* mehrfach erstellt werden, sondern es genügt mehrere Werte in einen gewünschten Werteeingang eines Knotens zu schicken.

In folgenden Abbildungen werden Quads einmal einzeln verteilt (siehe *img 1.3.11*), und anschließend mit Hilfe des LinearSpreads (siehe *img 1.3.12*).



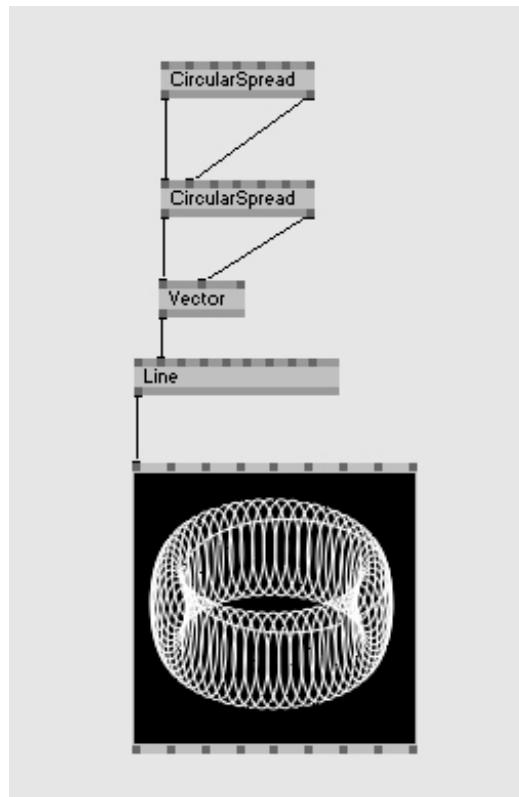
img 1.3.11



img 1.3.12

Bei den beiden Abbildungen wird deutlich, dass sich bei richtiger Anwendung von Spreaderzeugern die Gesamtmenge der Knoten deutlich reduzieren lässt.

Kombiniert man mehrere Spreaderzeuger, erreicht man schnell mit nur wenigen Knoten visuell komplexe Ergebnisse. In folgendem Beispiel (img 1.3.13) wurde das Quad durch eine Line ersetzt, die die Koordinaten aus dem Spreadgeber verbindet, und zwei CircularSpreads kombiniert. Das Ergebnis erinnert an Herbert W. Frankes Lissajous Figuren.



img 1.3.13

Obige Erläuterungen zeigen, wie durch die Anwendung einer Vielzahl von Koordinaten auf Primitives komplexe Visualisierungen entstehen. Darüberhinaus existiert eine Vielzahl an Programmier-Routinen, die sich generative GestalterInnen bedienen können. Eine Auswahl solcher Methoden wird in Kapitel 3 ausführlich behandelt.

1.4 Zusammenfassung

Dieses erste Kapitel hat den Begriff der *generativen Gestaltung* erläutert.

Die Abhandlung des Begriffs ist hiermit abgeschlossen.

Darauf basierend widmet sich das zweite Kapitel der Typografie.

Im dritten Kapitel wird Typografie mit der generativen Gestaltung zusammengeführt.

2. **Typografie**

2.1 Einleitung

Dieses Kapitel dient als Vorbereitung der Zusammenführung des Begriffs generative Gestaltung mit dem der Typografie.

Der Leser soll in diesem Abschnitt keine allgemeine Abhandlung über Typografie erwarten – der Begriff Typografie wird im Hinblick auf generative Gestaltungsmethoden untersucht.

Der erste Schritt ist eine Konkretisierung des Begriffes Typografie, wobei *Typografie* entsprechend der Thematik dieser Arbeit auf *Erstellung von Displayschriften* eingegrenzt wird.

Der zweiten Schritt ist eine technische Herleitung über Typografie-Darstellung am Computer. Der Prozess wird vom Aufbau von Schriftdateien bis zu ihrem Rendering am Bildschirm analysiert.

Darauf basierend werden Methoden zur Schriftdarstellung in VVVV vorgestellt und hinsichtlich generativer Eingriffe untersucht.

Am Ende des Kapitels ist der Begriff Typografie entsprechend eingegrenzt, und Basiswissen über den digitalen Prozess der Schriftdarstellung geschaffen.

2.2 Bedeutung von *Typografie* in dieser Arbeit

„Die Typografie bezeichnet sowohl die visuell gestalterische Organisation einer Schrift als auch die gestalterische und technische Realisation von Schriftzeichen.“

(Bilz 2004, 4)

Unter „visuell gestalterischer Organisation“ einer Schrift ist in diesem Zusammenhang die Tätigkeit zu verstehen, die auch unter dem Begriff Layouten bekannt ist. Das heißt, die Anwendung einer Schrift, mit entsprechenden Justagen wie der Positionierung, Absatzgestaltung, Zeilenabstände usw.

Die „gestalterische und technische Realisation“ meint das Erstellen einer Schrift: das Schriftdesign, sowie die technische Aufbereitung für deren Verwendung.

Eine detailliertere und erweiterte Klassifizierung nimmt Wolfgang Beinert in seinem Typolexikon vor:

„Typographie umfasst gegenwärtig:

- I. die Kulturwissenschaft und Lehre der historischen und neueren Schriftgeschichte, die Klassifikation von Druck- und Screenschriften sowie deren kunstgeschichtliche Zuordnung;
- II. das Wissen über Betrachtungs- und Lesegewohnheiten;
- III. die Lehre von der ästhetischen, künstlerischen und funktionalen Gestaltung von Buchstaben, Satzzeichen, Sonderzeichen und Schriften sowie deren Anwendungen in Druckwerken, in digitalen Medien und im dreidimensionalen Raum;
- IV. die Lehre, Sprache und Gedanken mittels maschinell bzw. digital reproduzierbarer Schriften sichtbar und den Anforderungen entsprechend optimal lesbar oder verständlich zu machen;
- V. die visuelle Gestaltung eines Druckerzeugnisses, einer Multi-Media-Arbeit oder einer dreidimensionalen Oberfläche in der Art, dass Inhalt und Schrift sowie die Anordnung von Text und Bild ein optisch und didaktisch befriedigendes Ganzes ergeben;
- VI. die Kenntnisse von der handwerklichen, druck- und programmtechnischen Implementierung einer Schriftsatzarbeit.“

(Beinert 2008a, 1)

Für diese Arbeit ist lediglich Beinerts dritter Punkt relevant: die Gestaltung von Schriftzeichen. Ziel ist, Schriftzeichen generativ zu gestalten und nicht, sie generativ gestalterisch zu organisieren.

Wie die geschichtliche Entwicklung zeigt, sind klassische Schriftarten sehr langlebig (vgl Willberg 2003, 20f). Die Gestaltung benötigt viel Zeit, um allen geforderten Faktoren, wie technische Kompatibilität und Lesbarkeit, gerecht zu werden.

„Schrift (...) zeigt sich in ihrer Formsprache untrennbar verknüpft zur gesellschaftlichen Entwicklung des Menschen. Schrift besteht aus einem Repertoire definierter Zeichen, deren Bedeutungsinhalte auf in Jahrhunderten geprägten Übereinkünften basieren. Das Erkennen und Identifizieren der Zeichen ist dabei Voraussetzung für das Funktionieren des Leseprozesses. Innerhalb eines Kulturkreises ist somit die Anatomie einer Schrift, die innere Struktur eines Buchstabens vorgegeben. So können bei unserem lateinischen Schriftalphabet Formänderungen einzelner Buchstaben nur innerhalb enger Grenzen stattfinden, um optimal entschlüsselt zu werden.“

(Bilz 2004, 4)

Schrift muss in erster Linie funktionieren und lesbar sein, sofern die Gattung der Textschriften betrachtet wird. Die klassischen Schriften sind allesamt den Textschriften zuzuordnen. Die Intention der frühen Schriftgestalter war es, Bücher und Texte als Träger von Inhalten zu verfassen und zu verbreiten. Die Gestaltung der Schrift hatte das Ziel, möglichst große Funktionalität zu erreichen:

„In der Gestaltung einer Textschrift sind die Spielregeln klar definiert und geben dem Schriftdesigner wenig Raum für Exzentrik. Übersicht und Zurückhaltung sind obligat, oder wie Adrian Frutiger es formuliert, wenn du Dich an die Form des Löffels erinnerst, mit welchem Du die Suppe gegessen hast, dann war es eine schlechte Form (...). Die Schrift muss so sein, dass der Leser sie nicht bemerkt (...). Eine gute Textschrift ist beides: banal und schön zugleich“.

(Bilz 2004, 64)

Generatives Gestalten neigt jedoch zu den im Zitat angesprochenen, exzentrischen Auswüchsen. Es ist, wie im ersten Kapitel gezeigt, zwar durchaus möglich feste Rahmenbedingungen für das generative System zu schaffen, jedoch existieren auch Parameter die von Benutzern verändert werden oder Zufallswerte sein können. Man muss davon ausgehen, dass Benutzer und Zufall exzentrische Launen besitzen. Die

Bildklasse des jeweiligen generativen Systems wird in solchen Fällen bis ins visuelle Extrem erweitert.

Eine Lesbarkeit von generativ erzeugten Schriftzeichen kann deshalb nicht hundertprozentig gewährleistet sein. Textschrift eignet sich demnach nicht für die Untersuchungen dieser Arbeit. Einen passenden Rahmen bildet die Displayschrift:

„Der Charakter einer Displayschrift ist einmalig, plakativ, bildhaft, ganzheitlich, provokativ und/oder expressiv. Spezifische Aufgabe einer Displayschrift ist, Präsenz zu demonstrieren. Der typografische Begriff „display face“ (...) bezeichnet generell große Schriften, die primär für Überschriften (Headlines), Titel und Schlagzeilen stehen oder zur Hervorhebung eingesetzt werden.“
(Bilz 2004, 14)

Der Begriff der Displayschrift zeigt sich also losgelöst vom Fließtext. In vielen Anwendungen bedeutet er Überschrift, jedoch kann Displayschrift auch als rein grafischer Bestandteil eines visuellen Produkts stehen und auf lesbare Inhalte völlig verzichten. In vielen Fällen wird Displayschrift verwendet um expressiv Stimmungen in den Vordergrund zu stellen, was durchaus auf Kosten der Lesbarkeit passieren kann, die Aussage jedoch nicht abschwächt (vgl Bilz 2004, 14).

Die Klasse der Displayschriften funktioniert besser im generativen Experimentierfeld als die Textschriften.

Im weiteren Verlauf wird sich, hinsichtlich dem Typografiebegriff, auf die *Gestaltung von Displayschriften* bezogen.

2.3 **Digitale Typografie**

Generative Gestaltungsstrategien beziehen sich in dieser Arbeit auf Gestaltung am Computer. Dies führt zur Notwendigkeit, den Begriff der Typografie und die informationstechnischen Prozesse ihrer digitalen Anwendung aufzuzeigen.

Zunächst wird die Schrifterzeugung in Windows und die dafür gängigen Schriftformate untersucht.

Die Entscheidung das Forschungsfeld auf Windows einzugrenzen, basiert auf der Tatsache, dass die im weiteren betrachtete Programmierumgebung VVVV auf Windows basiert. Viele der aufgezeigten Fakten gelten dennoch auch für andere Betriebssysteme, da sich seit Einführung des OpenType Formats die Handhabung von Schriften in beiden Systemen angenähert hat.

Weiters werden Möglichkeiten zur Schrifterzeugung in VVVV aufgezeigt. Dabei wird auf drei unterschiedliche Methoden eingegangen und untersucht, ob diese hinsichtlich generativer Eingriffe geeignet sind.

2.3.1 Schrift in Windows – von der Schriftdatei zur Bildschirmausgabe

Um in Windows Schrift auf den Bildschirm zu bringen, sind mehrere informationstechnische Schritte notwendig. Die Beschaffenheit von Fontdateien wird zunächst als Basisinformation vorangestellt.

Im Jahre 1999 wurde von Microsoft und Adobe das OpenType Format eingeführt (vgl *Microsoft 2001*). Dadurch sollte das vorhergehende Windows TrueType Format, zusammen mit dem auf Druck fokussierten Postscript Format vereinigt werden, und plattformübergreifende Kompatibilität erreicht werden.

„The OpenType font format addresses the following goals:

- broader multi-platform support
- better support for international character sets
- better protection for font data
- smaller file sizes to make font distribution more efficient
- broader support for advanced typographic control“

(Microsoft 2001)

Diese Ziele sind erreicht worden, und OpenType hat sich aufgrund dieser Vorteile mittlerweile etabliert.

Auch wenn im Folgenden teilweise Eigenschaften des TrueType Formats herbeigezogen werden, sind diese Erläuterungen auch für das OpenType Format gültig. OpenType ist die direkte Erweiterung von TrueType – die Basisprinzipien sind dieselben.

Was ist der Inhalt einer Font Datei?

„One of the more obvious things TrueType fonts include is the shape of each character. Each and every letterform contained in a TrueType font is stored as an outline, or more accurately, as a mathematical description of the character constructed from a series of points. For this reason, TrueType is known as an outline font format.“

(Microsoft 1997b)

Die Schriftgeometrie liegt in Form von Bézierkurven vor. Die Vorteile von diesen so genannten Vektordaten wurden bereits in Kaptitel 1.3.1 dargelegt.

Für die Darstellung von TrueType Schriften am Bildschirm, wird folgender Prozess durchlaufen (vgl Microsoft 1997):

- Outlinekoordinaten der Font ins Koordinatensystem zeichnen
- Die Form entsprechend dem BenutzerInnenwunsch entsprechend skalieren
- Die Outline auf Größe des Pixelkoordinatensystems umrechnen und anpassen
- Unter Umständen buchstabenspezifische Sonderregeln für bestimmte Größen anwenden
- Berechnen, welche Pixel den Buchstaben bilden und diese füllen
- Bitmap auf Rasterausgabe (Bildschirm) rendern

TrueType SchriftgestalterInnen legen beim Entwerfen der Schrift Koordinaten der Bézierinformationen fest. Diese können entweder Linien sein, die jeweils durch zwei Punkte definiert werden oder Kurven, die durch zwei Ankerpunkte und zwei Kontrollpunkte beschrieben werden (vgl Microsoft 1997).

Für jeden Buchstaben verfügt die GestalterIn über die Fläche von einem *em square*. In diesem Quadrat, das noch von den Bleilettern Gutenbergs kommt, kann die Auflösung frei gewählt werden. Es beschränkt sich auf einen Bereich von -16384 bis +16384 so genannten FUnits. FUnits oder FontUnits sind eine relative Größenangabe, die das Koordinatensystem für jeden Buchstaben auflösen.

„An em square is exactly 9 points high when a glyph is displayed at 9 points, exactly 10 points high when the font is displayed at 10 point, and so on.“
(Microsoft 1997)

Diese Koordinaten werden beim Setzen der Schrift auf die gewünschte Größe skaliert und auf die absolute Pixelgröße umgerechnet. Dabei ist zu beachten, dass verschiedene Ausgabemedien verschieden fein aufgelöst sind. Das heißt, eine 9 Punkt Schrift hat eine variierende Pixelanzahl, je nachdem auf welchem Medium sie dargestellt wird. Ein VGA Bildschirm unter Windows hat eine Auflösung von circa 96 dpi, das bedeutet eine 96 dots per inch (Pixel pro Inch) Einteilung. Bei einem Druck ist diese Zahl deutlich höher (ab 300 dpi bei Laserdruckern) (vgl Microsoft 1997), bei LED Wänden hingegen sehr niedrig (teilweise nur ein Pixel pro Zentimeter).

Die letzten Schritte um den Buchstaben auf dem Bildschirm darzustellen bestehen darin, die Vektordaten in Pixeldaten umzuwandeln. Der Scanconverter scannt das *em square* Pixel für Pixel ab und prüft, ob sich das Pixel jeweils innerhalb oder außerhalb der Vektorform des Schriftzeichens befindet (vgl Microsoft 1997).

„The TrueType scan converter takes an outline description of a glyph and produces a bitmap image for that glyph. (...)“

Rule 1: If a pixel's center falls within the glyph outline, that pixel is turned on and becomes part of that glyph.

*Rule 2: If a contour falls exactly on a pixel's center, that pixel is turned on.“
(Microsoft 1997)*

Die meisten Anwendungen bieten darüberhinaus Methoden zur Kantenglättung an. Diese zusätzlichen Regeln des Scanconverters füllen Pixel, die am Rande der Schriftzeichen-Kontur liegen in transparenten Abstufungen. Dieses Antialiasing genannte Verfahren, sorgt für bessere Lesbarkeit auf Bildschirmen.

2.3.2

Schrift in VVVV

Der Windows eigene Workflow wird im folgenden von der Software VVVV kontrolliert und an verschiedenen Stellen überschrieben.

Welche Möglichkeiten VVVV bietet, um Schrift darzustellen und an welcher Stelle der Standard-Windowsprozess unterbrochen, und die Kontrolle den generativen GestalterInnen übergeben wird, wird hier untersucht.

An dieser Stelle ist zu beachten, dass die erläuternden Abbildungen dieses Kapitels nicht als generative Gestaltungsergebnisse einzustufen sind, obwohl das generative Tool VVVV verwendet wird. Die Knoten, die die Schrift darstellen bringen alleine keine visuell ausreichende Komplexität hervor, um von einer gestalteten Bildklasse und einem generativen System sprechen zu können.

2.3.2.1 Schrift im GDI Renderer

Die einfachste Möglichkeit Schrift in VVVV zu erzeugen bietet der *Text (GDI)* Knoten, welcher Microsofts GDI API für den Renderoutput benutzt. Dementsprechend wird in diesem Fall nicht der in Kapitel 1.3.2 erwähnte *Renderer (EX9)* sondern der *Renderer (GDI)* benutzt.

„The Microsoft® Windows® graphics device interface (GDI) enables applications to use graphics and formatted text on both the video display and the printer. Windows-based applications do not access the graphics hardware directly. Instead, GDI interacts with device drivers on behalf of applications.“
(Microsoft 2008)

Der große Nachteil an der Tatsache, dass der Grafikchip (GPU) nicht direkt für Berechnungen angesprochen werden kann ist, ist dass der Hauptprozessor (CPU) die ganze Leistung alleine aufbringen muss. Bei komplexen Patches stößt man schnell an die Hardwaregrenzen. Darüberhinaus ist die GDI Grafikbibliothek nicht für aufwändige 3d Umgebungen konzipiert, sondern für simple, informative Darstellungen. GDI dient meist dafür, Werte grafisch zweidimensional zu repräsentieren (in ein Koordinatensystemen zeichnen). Eine Tiefeninformation, also ein echter 3d Raum existiert in der GDI Umgebung nicht.

In VVVV ist der GDI Renderer hauptsächlich in Helpatches zu finden, da bereits in einer kleinen Applikation die Funktion von Knoten visualisiert werden kann.

Die Textfunktionen der GDI API sind hingegen sehr vielseitig. Abbildung *img 2.3.01* zeigt, dass der *Text (GDI)* Knoten über zahlreiche Eingangspins, die den darzustellenden Text genau festlegen lassen, verfügt. Vor allem die Optionen der Textausrichtung und der Zeilenumbrüche sind ein Vorteil gegenüber den anderen Methoden. *Text (GDI)* ist die sauberste und dem normalen Windows Workflow am direktesten entsprechende Methode, Text in VVVV zu erzeugen.



img 2.3.01

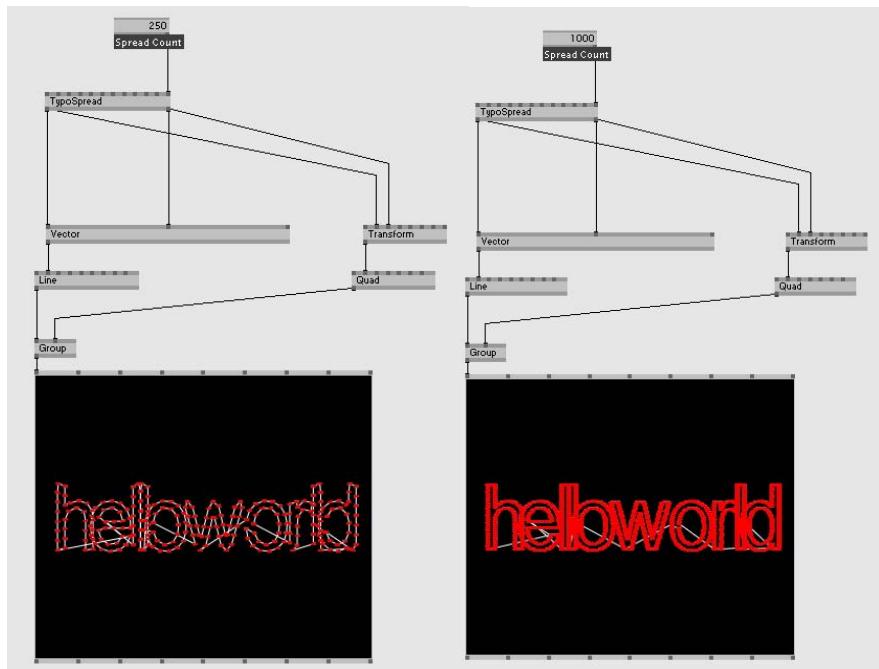
Für weitere generative Untersuchungen dieser Arbeit wird der *Text (GDI)* Knoten jedoch ausgeklammert, da er aufgrund der Nachteile eine sehr statische, textabbildende Funktion einnimmt. Für visuelle Experimente ist er weitgehend ungeeignet.

2.3.2.2 Typospread

Der in Kapitel 1.3.2 bereits erwähnte *Typospread (Spreads)* ist in erster Linie ein Wertelisten-Erzeuger. Die Ausgabepins liefern keine fertige graphische Ebene (Layer) wie beim *Text (GDI)*, sondern X und Y Koordinaten, die sich auf der Outline der Schrift befinden.

Das macht ihn unabhängig vom verwendeten Renderer – die Koordinaten können sowohl auf GDI als auch auf DirectX Elemente angewendet werden. In den folgenden Beispielen werden die X/Y Koordinaten auf die X/Y Position der Objekte angewendet, da dies zu lesbaren Ergebnissen führt.

Folgende Abbildung (img 2.3.02) zeigt den Typospread, angewendet auf eine *Line (EX9.Geometry)* die die ausgegebenen Koordinaten miteinander verbindet und auf ein *Quad (DX9)*, das sich auf den Punkten platziert. Die Anzahl der Punkte kann mit Hilfe des SpreadCount Pins am Typospread festgelegt werden. Je höher diese Zahl ist, desto lesbarer wird das Ergebnis.



img 2.3.02

Die Nachteile des Typospreads sind, dass er nicht geeignet ist präzise Schrift zu erstellen. Durch die Positionierung der Punkte auf der Schriftoutline wird nie die darstellerische Qualität erreicht, die mit gewöhnlichen Anwendungen und dem Windows Standardprozess möglich ist. Auch bei hoher Punkteanzahl (SpreadCount), handelt es sich nur um eine Annäherung an die Vektordaten.

Darüberhinaus ist es nicht möglich, die so erzeugte Schrift zu füllen. Der Typospread Knoten gibt lediglich X/Y Werte aus und keine Information darüber, welche dieser Punkte eine geschlossene Formen bilden sollen. Das wird sichtbar, wenn die Koordinaten mit einer Linie verbunden werden, und sich diese buchstabenübergreifend fortsetzt (siehe *img 3.2.02*). Die Unterscheidung, nach wievielen Punkten der nächste Buchstabe beginnt, ist nicht ohne weiteres möglich.

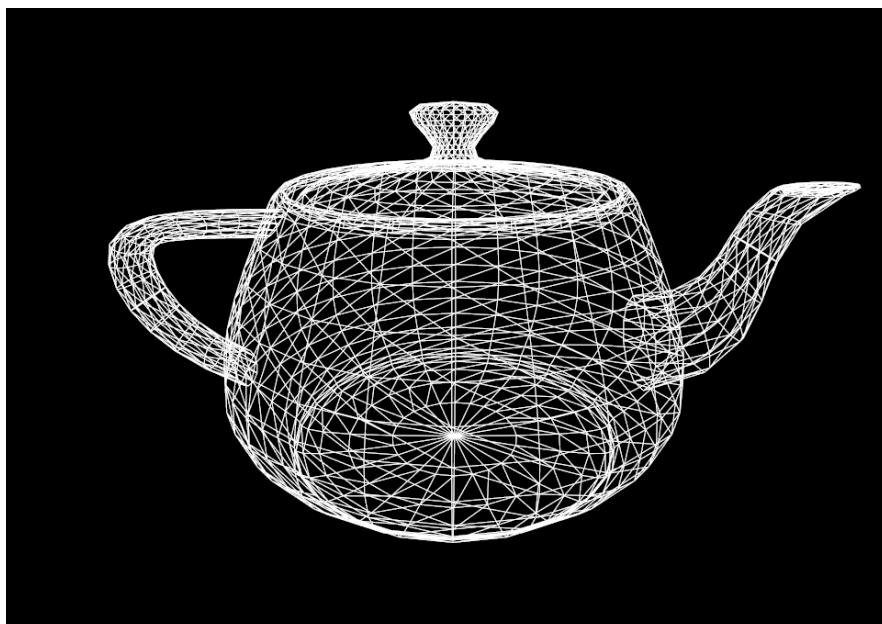
Dennoch ist der Typospread ein geeigneter Knoten, um generativ in Schrift einzugreifen. Die oben genannten Negativkriterien sind oft kein Hindernis, wenn es um die Erstellung von Displayschriften geht.

Der Typospread liefert die Grundbestandteile von Schrift in Zahlenform. Diese Koordinaten sind für programmiererische Eingriffe besonders geeignet, da das Einfügen eigener Algorithmen an dieser Stelle besonders einfach ist.

2.3.2.3 Schrift als 3d Objekt

Eine dritte Möglichkeit Schrift in VVVV zu generieren, bietet der Knoten *Text (EX9.Geometry)*. Dieser gibt eine DirectX Geometrie aus.

Geometrien werden in VVVV als Polygonnetz (auch: Mesh) ausgegeben. Dabei handelt es sich um einen gängigen Standard bei der Erzeugung und Bearbeitung von 3d Grafiken, das die Oberflächen von dreidimensionalen Objekten aus einer Zusammensetzung von Polygonen (in VVVV: Dreiecken) beschreibt.

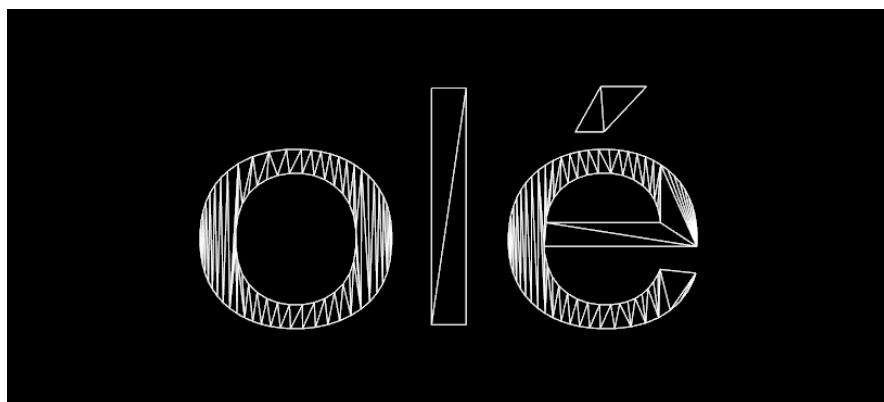


img 2.3.03

img 2.3.03 zeigt den bekannten Teapot¹ als Polygonnetz. Diese Ansicht wird auch Wireframe Darstellung genannt.

Die Beschreibung eines 3d Objekts mit Hilfe eines Polygonnetzes hat den Nachteil, dass keine echten Rundungen existieren. Es sind lediglich Annäherungen an runde Oberflächen, durch ein hohes Auflösen des Netzes möglich. Dadurch ist bei Schrift die Komplexität der 3d Information von einzelnen Buchstaben sehr unterschiedlich, wie *img 2.3.04* zeigt.

¹ Der Teapot wird seit den Anfängen der 3d Grafik als Standardobjekt verwendet, um verschiedene Techniken zu demonstrieren.



img 2.3.04

3d Geometrien bestehen aus einer Liste von Koordinaten ihrer Polygoneckpunkte², und können von VVVV nicht direkt gezeichnet werden. Zwischen Geometrie und Renderer muss ein so genannter Shader („Schattierer“), der die Darstellung der Koordinaten im Raum festlegt. Shader können mindestens die Transformation und die Farbe des Objekts definieren oder das Objekt mit Lichtquellen physikalisch korrekt beleuchten³.

Shader in VVVV sind .fx Dateien, die selbst hinzugefügt oder modifiziert werden können. Voraussetzung dafür ist die Beherrschung der High Level Shader Language der DirectX API.

² Plus Koordinaten der zugehörigen Normalen. Diese sind für weitere Betrachtung in dieser Arbeit jedoch irrelevant.

³ Dies gilt zumindest für so genannte Pixelshader. Vertexshader sind darüberhinaus in der Lage komplexe Transformationen des Objekts direkt auf der Grafikkarte durchzuführen, was eine enorme Entlastung die den Prozessor bedeutet. Dies ist heute eine gängige Praxis in VVVV, besitzt hinsichtlich Typografie jedoch keine Relevanz.

Im der Abbildung *img 2.3.05* werden verschiedene, in VVVV standardmäßig implementierte Shaderarten gezeigt:

links oben: Constant, rechts oben: Flat Point

links unten: Gouraud Point, rechts unten: Phong Directional

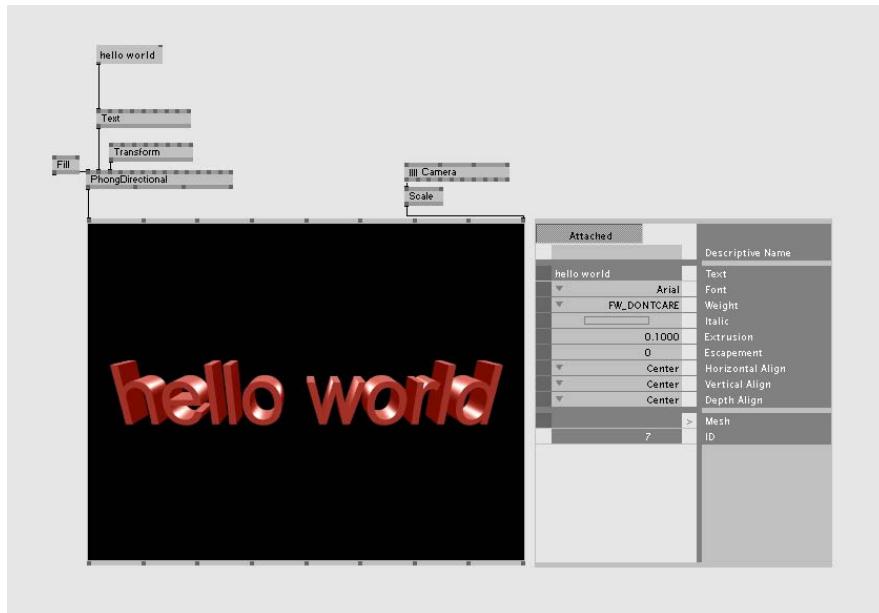
Die Bezeichnungen Point und Directional beschreiben das Licht: es wird zwischen Punktlicht und Richtungslicht unterschieden.



img 2.3.05

Um zurück auf den *Text (EX9.Geometry)* Knoten zu kommen, wird im folgenden betrachtet, welche Funktionen er bietet, um eine Font als 3d Objekt darzustellen.

Ein Blick auf die Einstellungsmöglichkeiten des Knotens zeigt das Konzept: Es existieren bekannte Einstellungsmöglichkeiten für die Darstellung der zweidimensionalen Schrift, sowie 3d Parameter. Davon ist *Extrusion* der dreidimensional entscheidende, denn dieser gibt an, wie weit sich die Schrift in die Tiefe streckt.



img 2.3.06

Die Schrift wird von Fontkoordinaten in Polygonkoordinaten umgewandelt, und im Anschluss nach Belieben in der z-Achse skaliert. Somit erhält man statt einem flachen Pixelrendering, ein dreidimensionales Objekt, das auch im virtuellen Raum funktioniert.

In VVVV existieren eine Vielzahl von Möglichkeiten, 3d Objekte in Szene zu setzen, die sich auch für die aus Text erzeugten 3d Objekte nutzen lassen. Positionierung und Bewegung im Raum bis hin zu komplexen Verzerrungen des Polygonnetzes sind in VVVV möglich.

Der Nachteil der Schrifterzeugung mit dem Text (Ex9.Geometry) Knoten ist der Nachteil des Polygonnetzprinzips: die Unfähigkeit, echte Rundungen darzustellen.

Aufgrund der enormen Möglichkeiten 3d Objekte zu behandeln, ist 3d Text in VVVV eine probate Methode für Experimente mit generativer Typografie.

Mit Hilfe zusätzlicher Knoten kann, ähnlich wie beim Typospread, auf Koordinaten des Polygonnetzes in Zahlenform zugegriffen werden, was zusätzliche algorithmische Eingriffe ermöglicht.

Zusammenfassend ist deutlich geworden, dass VVVV eine Vielzahl von komplett unterschiedlichen Methoden bietet, Schrift zu erzeugen. Diese beziehen sich stets auf den Windows Workflow, aber brechen an verschiedenen Stellen ab und lassen die BenutzerInnen bzw. VVVV

entscheiden, was mit den Koordinaten aus der Fontdatei passieren soll.

Eine Grundwerkzeugpalette ist mit dem Typospread und dem 3d Text an dieser Stelle gelegt und der Weg geebnet, generative Eingriffe an Schriften vorzunehmen.

3. **Methoden**

Einleitung

Dieses dritte Kapitel hat die Aufgabe, hinsichtlich der in der Einleitung formulierten Forschungsfrage zu untersuchen, welche Methoden generativer Gestaltung für die Erstellung von Displayschriften geeignet sind. Dabei ist zu beachten dass sich die Definition von generativer Gestaltung in dieser Arbeit, wie im ersten Kapitel hergeleitet, nicht nur auf die technische Basis bezieht, sondern, nach John Maeda, auch einen spezifischen ästhetischen Anspruch hat.

Das hier folgende Kapitel widmet sich genau dieser Thematik: Welche Methoden zur Schriftdarstellung in VVVV lassen sich mit welchen generativen Gestaltungsmethoden verknüpfen, um Bildklassen von Displayschriften zu erstellen, mit dem vorausgesetzten, gestalterischen Mehrwert?

Der Begriff generative Gestaltungsmethoden meint in diesem Zusammenhang Programmier-Routinen, die häufig für generatives Gestalten herangezogen werden. Es existiert eine Vielzahl verschiedener Standardalgorithmen – jedoch sind nur einige davon geeignet, um Gestaltung zu generieren.

Der Autor wählt im folgenden drei gängige Methoden aus diesem Pool aus und untersucht sie hinsichtlich ihrer Tauglichkeit, Schrift zu erzeugen bzw auf VVVVs Schrifterzeuger einzuwirken.

Ein vollständige Auflistung dieser Methoden existiert nicht. Einen Überblick bietet aber einerseits „Code@Art“ (siehe *Trogemann/Viehoff 2005*), worin Programmiermethoden künstlerischer Praxis beschrieben werden, die sich stellenweise ins Design transferieren lassen. Andererseits erläutert das Buch „Processing. A Programming Handbook for Visual Designers and Artists“ (siehe *Fry/Reas 2007*) Methoden generativer Gestaltung, jedoch bezogen auf die Programmierumgebung Processing.

Aus Verständnisgründen werden in der anfänglichen Betrachtung einer Methode stets visuell reduzierte Abbildungen gezeigt. So wird versucht, das gestalterische Potenzial der jeweiligen Strategie zu analysieren. Am Ende des jeweiligen Unterkapitels werden ausgearbeitetere Beispielen gezeigt.

3.2 Zufall

3.2.1 Hinführung

Zufall ist eine beliebte Methode, externe Parameter die außerhalb der Kontrolle der GestalterInnen liegen, in ein Gestaltungsprodukt einfließen zu lassen.

Um dieses Prinzip anwenden zu können, wird vorweg kurz erörtert, wie Computer Zufälle zu erzeugen.

„Randomness is a lack of order, purpose, cause, or predictability. A random process is a repeating process whose outcomes follow no describable deterministic pattern, but follow a probability distribution.“

(Wikipedia 2008c)

Dies stellt eine allgemeine Definition des Zufalls dar. Inwiefern diese beim generativen Gestalten relevant ist, zeigen die folgenden Ausführungen.

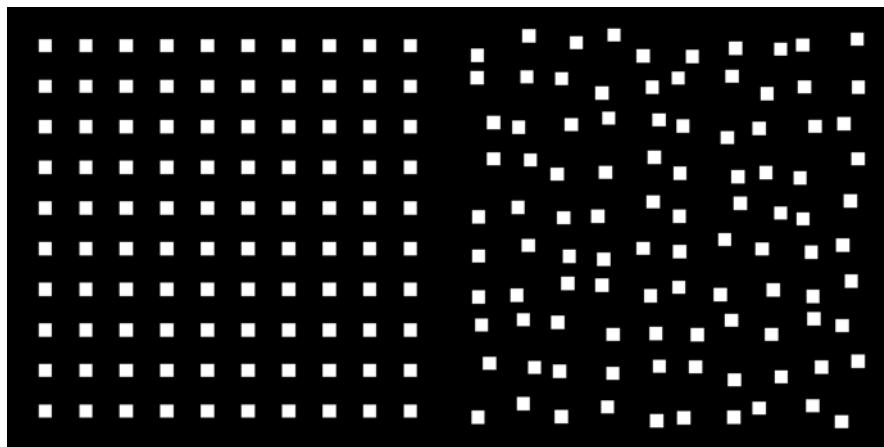
Den Zufall in Gestaltung und Kunst einfließen zu lassen, hat spätestens seit der modernen Kunst Tradition. Künstler wie Marcel Duchamp, Jean Arp und John Cage sind bekannt für ihre Experimente mit dem Faktor Zufall (vgl Fry 2007, 127).

„Actions like dropping, throwing, rolling etc., deprive the artists of certain aspects of decisions. The world's chaos can be channeled into making images and objects with physical media. In contrast, computers are machines that make consistant and accurate calculations and must therefore simulate random numbers to approximate the kind of chance operations used in nondigital art.“

(Fry 2007, 127)

Das weltliche Chaos oder milder ausgedrückt, natürliche Unregelmäßigkeiten, sollen also mit Hilfe von Zufallszahlen in digitale Gestaltung eingeführt werden.

Folgende Abbildung (*img 3.2.01*) zeigt eine Möglichkeit des Einflusses von Zufallszahlen auf die Anordnung von Rechtecken.



img 3.2.01

Eine unregelmäßige Verteilung von Objekten ist mit Hilfe von Zufallszahlen mit geringem Aufwand zu erreichen.

„Zufall spielt in den Naturwissenschaften eine bedeutende Rolle. Die am genauesten bekannten Naturkonstanten sind Ergebnisse physikalischer Zufallsprozesse. So wundert es nicht, dass eine Vielzahl von Computeranwendungen, zum Beispiel mathematische Optimierungsverfahren, Simulationsprozesse, Spiele, genetische Algorithmen und kryptographische Methoden, Zufallswerte benötigen, um Realitätsnähe zu gewährleisten.“

(Trogemann/Viehoff 2005, 354)

3.2.2 Das Prinzip Pseudozufall

Diesen natürlichen Zufall, der meist auf physikalischen Begebenheiten beruht in maschinelle und digitale Prozesse einzubinden, ist nicht möglich.

„Die ganze Ingenieurskunst wird darauf verwendet, die Genauigkeit, Zuverlässigkeit und beliebige Wiederholbarkeit aller Berechnungen und Symbolmanipulationen – in der Fachsprache: die ‚Determiniertheit der Maschine‘ – sicher zu stellen.“

(Trogemann/Viehoff 2005, 346)

Determiniertheit tritt auch in der Algorithmusdefinition aus Kapitel 1.1.1 auf. Die Determiniertheit meint die Eigenschaft, in einem System ein bei gleichen Startbedingungen reproduzierbares Ergebnis hervorzubringen. Dies stellt einen Widerspruch zur allgemeinen Zufallsdefinition dar.

Die Annäherung an zufällige Ergebnisse mit Hilfe von Algorithmen nennt man deshalb Pseudozufallszahlen (vgl Trogemann/Viehoff 2005, 349).

„Von ‚echten‘ Zufallszahlen spricht man, wenn die Werte auf der Basis realer physikalischer Prozesse gewonnen werden. Als Standardbeispiel wird an dieser Stelle immer das Werfen eines Würfels genannt, der als mögliche Ergebnisse die Zufallszahlen 1, 2, ..., 6 liefert. (...) Solche Zufallszahlen können nur realisiert werden indem externe, d.h. nicht von der deterministischen Maschine erzeugte Ereignisse in den Verarbeitungsprozess integriert werden.“

(Trogemann/Viehoff 2005, 354)

Ein Pseudozufall wird mit Hilfe komplex verschachtelter Regelwerke erzeugt. Dabei kann so hohe Komplexität erreicht werden, dass keine analytische Bestimmung des Verhaltens mehr möglich ist (vgl Trogemann/Viehoff 2005, 349).

Diese so erzeugten Zahlen sind für gestalterische Zwecke unvorhersehbar und komplex genug, um sie als zufällig zu bezeichnen. Im Folgenden wird sich stets auf Pseudozufallszahlen bezogen.

3.2.3 Zufallsgeneratoren in VVVV

VVVV bietet zwei verschiedene Zufallsgeneratoren: *Random (Value)* und *RandomSpread (Spreads)*.

Der *Random (Value)* Knoten erzeugt im aktvierten Zustand jeden VVVV Frame einen neuen Zufallswert. Die Wertespanne ist skalierbar.

Der *RandomSpread* ist ein Spreaderzeuger der bereits im *Kapitel 1.3.2.4* vorgestellt wurde. Er verfügt über ähnliche Einstellungsmöglichkeiten wie der *LinearSpread*: einem Werte-Mittelpunkt, einem Wertebereich und einer Wertanzahl.

Zusätzlich existiert der Pin *RandomSeed*, an dessen Beispiel das Prinzip der Pseudozufallszahlen gut veranschaulicht werden kann:

Ein neu erstellter *RandomSpread* gibt immer genau den Wert -0.4686 aus. Das liegt einerseits an den Standardparametern, die beim Erstellen des Knotens geladen werden und andererseits am Prinzip Pseudozufall:

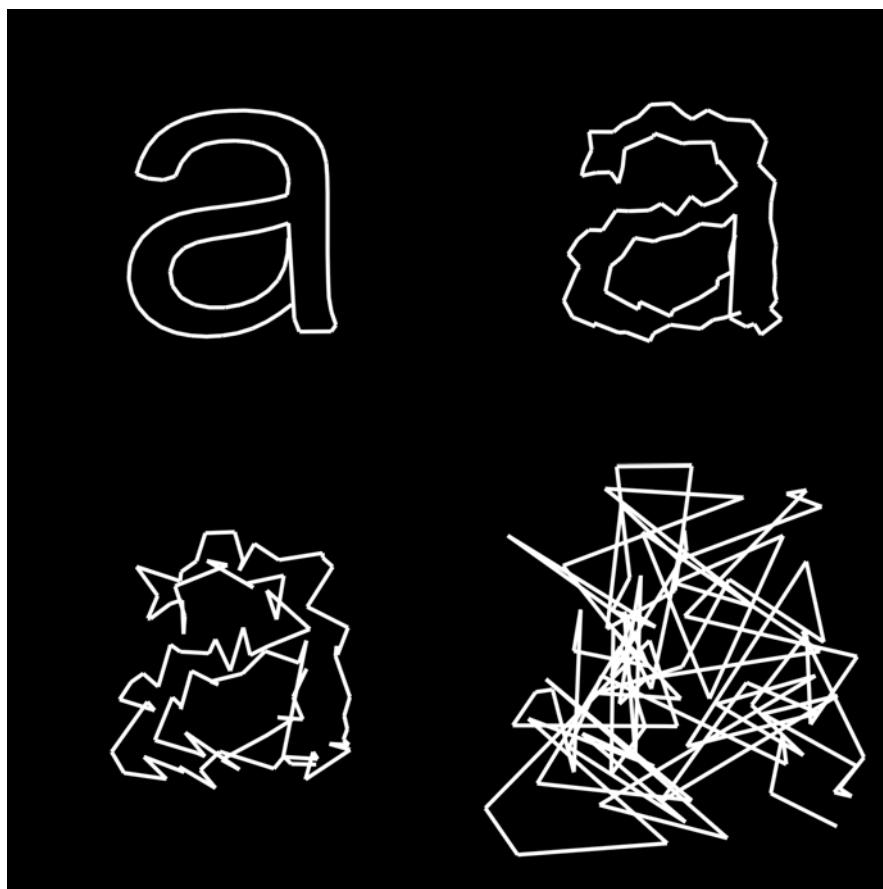
„the values generated are fully reproducible and are not random in any sense. You can build your patches on the fact, that each random spread with the same seed will generate exactly the same values on all machines.“
(VVVV.org 2008c)

Der RandomSeed legt dabei den internen Status des Pseudozufallszahlen-Generators fest. VVVV ermöglicht der BenutzerIn, den Status zu entscheiden oder durch weitere Algorithmen entscheiden zu lassen.

3.2.4 Typografische Anwendungen

Zufall sorgt für Unregelmäßigkeit bis hin zum Chaos, sofern die GestalterIn ihn nicht ausreichend formt. Dieses Formen ist für den Gestaltungsprozess der Typografie attraktiv, da sich SchriftgestalterInnen stufenlos zwischen Lesbarkeit und Unlesbarkeit bewegen können. Der Zufall ist frei dosierbar.

Für eine anfängliche typografische Untersuchung wird der Zufall benutzt, um die Form von fertigen Glyphen zu beeinflussen. Dazu wird in folgenden Demonstrationen der Typospread benutzt, der es ermöglicht, Punkte auf den Outlines der Buchstaben in ihrer Position zufällig zu verändern.



img 3.2.02

Die Abbildung *img 3.1.02* zeigt, wie der Zufall vom Ausgangsstadium des Buchstabens bis hin zur Unlesbarkeit skaliert wird.

Im nächsten Beispiel werden nicht die x- und y-Koordinaten sondern die z-Koordinaten durch Zufallszahlen beeinflusst.

Die Veränderung ist bei verzerrungsfreier Perspektive erst nach Drehung des Objektes erkennbar, wie Abbildung *img 3.2.03* zeigt.



img 3.2.03

Ein möglicher Einsatzbereich für dieses Verfahren wäre ein typografisches Musikvideo, das anfangs aus einem Liniengewirr besteht und nach und nach lesbar wird. Dieser Effekt kann einerseits durch Skalierung des Zufalls (*img 3.2.02*) und andererseits durch Drehen der Kamera (*img 3.2.03*) erzielt werden.

Als Zwischenergebnis bleibt festzuhalten, dass bei zufälligem Verändern von Punkten auf Schriftoutlines bereits generativ erzeugte Displayschriften, nach gültiger Definition, erstellt werden können. Die notwendige gestalterische Komplexität der Bildklasse ist gegeben. Darüberhinaus lässt sich der Trend feststellen, dass sich seriflose Schriften besser eignen. Dies ist darauf begründet, dass sich reduzierte Zeichen leichter identifizieren lassen, auch wenn sie verformt werden.

Eine zweite mögliche Anwendung des Zufalls, die sich für typografischen Einsatz anbietet, wird im folgenden aufgezeigt:
Anstatt die Positionen der Punkte zu verändern, werden zufällige Punkte, aus der Punktemenge die den Buchstaben beschreibt, ausge-

wählt und miteinander verbunden. Es befinden sich alle Punkte auf ihren richtigen Positionen, jedoch werden zufällig ausgewählte, Punkte mit einer Linie verbunden. So kann eine sehr große Zahl von verschiedenen, abstrakten Buchstaben erzeugt werden (siehe *img 3.2.04*).



img 3.2.04

Zusätzlich ist die Abstufung der Abstraktion frei wählbar. Von wenig bis zur vollen Punktezahl kann beliebig gewählt werden, wie reduziert oder auch *kaputt* der Buchstabe aussehen soll (siehe *img 3.2.05*).



img 3.2.05

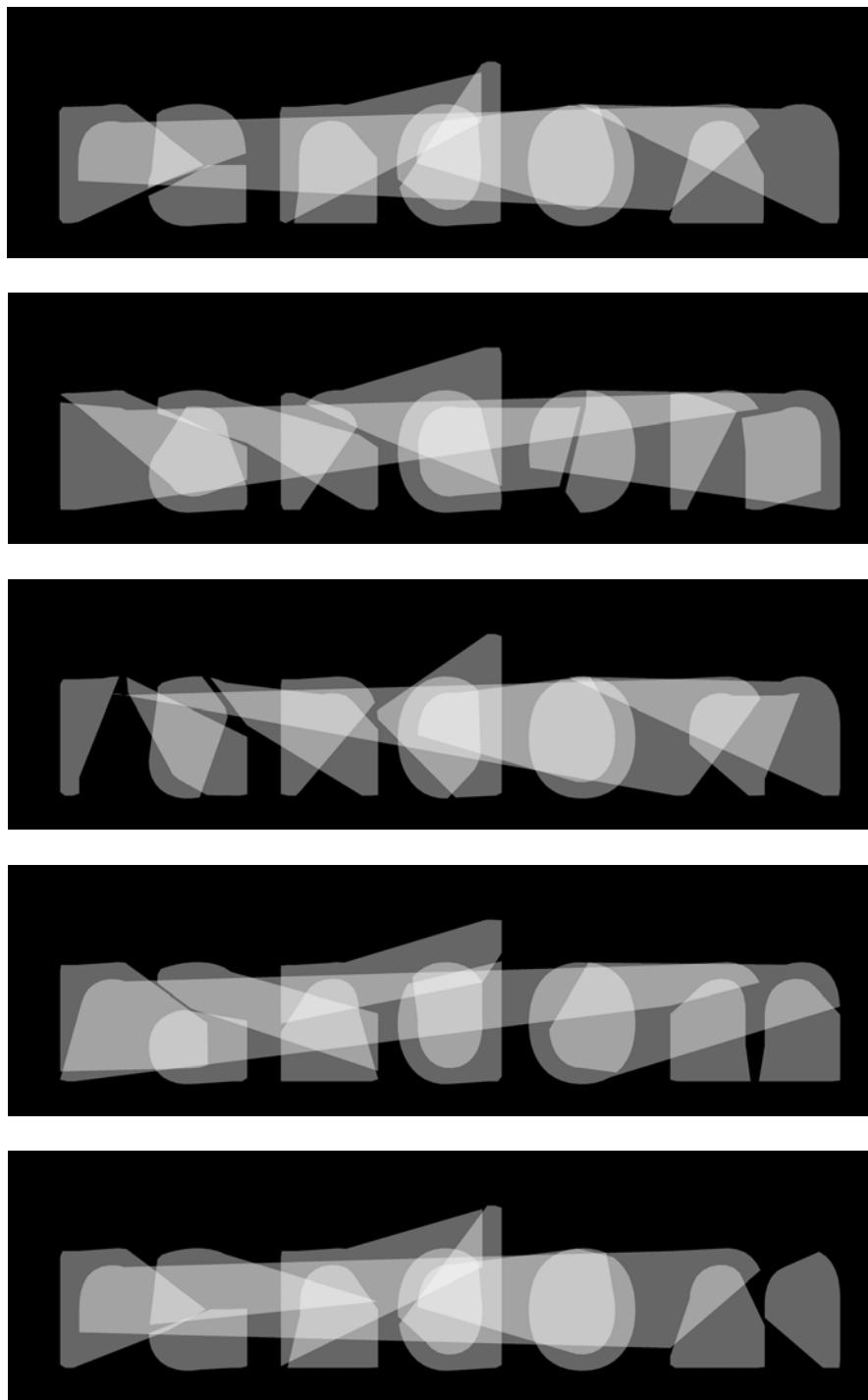
Wird nun die Generativität hinsichtlich dem Begriff der Bildklasse betrachtet, zeigt sich an den hier vorgestellten Anwendungen des Zufalls, dass diese generative Gestaltungsmethode für die Erzeugung von Displayschriften sehr gut geeignet ist. Wie *img 3.2.04* verdeutlicht kann die Bildklasse zum Beispiel aus Zufallsvariationen bestehen und/ oder aus Abstufungen in der Abstraktion (*img 3.2.05*).

Zum Abschluss des Kapitels werden gestalterisch weiter ausgearbeitete Beispiele gezeigt.



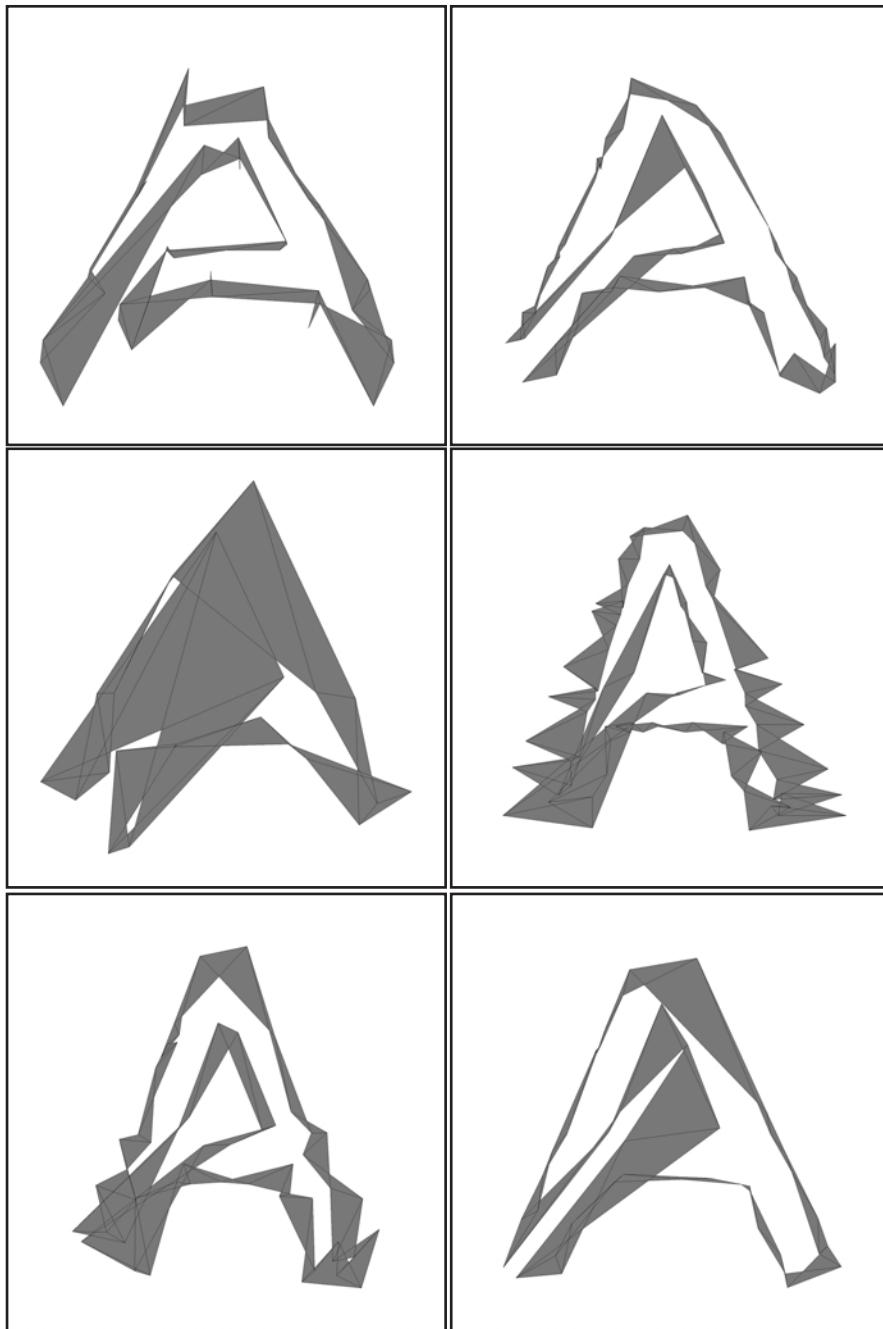
img 3.2.06

In Abbildung *img 3.2.06* wurden nicht wie in vorhergehenden Beispielen Zufallskoordinaten zu den Koordinaten des TypoSpreads addiert, sondern auf die Vertices eines Ex9 Texts. Daraus ergibt sich ein zerstört anmutendes 3d-Modell eines Buchstabens.



img 3.2.07

Abbildung *img 3.2.07* zeigt eine komplexere Verwendung von Zufallszahlen: Auf dem Typospread basierend werden mit Hilfe von Flächen teilungs-Algorithmen einzelne Fragmente eines Wortes in zufälligen Varianten miteinander zu Flächen verbunden.



img 3.2.08

Abbildung *img 3.2.08* zeigt eine komplexe Version des in *img 3.2.06* gezeigten Prinzips: dem Weglassen zufälliger Koordinaten. Der Effekt wurde durch weitere Zufallszahlen (Auflösung) verstärkt und auf ein dreidimensionales Band angewendet, das sich um die Form des Buchstabens legt.

3.3 Partikelsysteme

3.3.1 Hinführung

Als zweite Methode wird das Animationsverfahren Partikelsystem betrachtet.

Ein Partikelsystem wird zur Visualisierung von zB Feuer, Rauch, Staub oder für Fischschwärme und Herdenverhalten verwendet. Mit Hilfe von gekoppelten Massenpunkten werden solche Phänomene simuliert (vgl Trogemann/Viehoff 2005, 396).

Dabei wird nicht ein einzelnes 3d-Objekt animiert bzw verformt, sondern ein physikalisches Regelwerk gestaltet, welches mehrere Teil-Objekte animiert. Die Summe dieser Objekte ergibt daraufhin die Visualisierung des gewünschten Phänomens.

Das Verfahren wurde populär durch den Einsatz in Star Trek II im Jahre 1982 (siehe *img 3.3.01*):

„Der Grafikspezialist William T. Reeves (...) entwickelte für den Kinofilm 'Star Trek II: The Wrath of Khan' (...) einen realitätsnahen Explosions- und Feuerefekt. Grundlegendes Schema dieses Effektes war die Darstellung der Feuersbrunst durch zahlreiche lichtemittierende Kleinpartikel, die durch ein zentrales System ausgestoßen wurden und sich entlang von im System festgelegten physikalischen Gesetzen bewegten.“

(Hein 2002, 3)



img 3.3.01

3.3.2 Funktion von Partikelsystemen

Als erster Schritt wird die Beschaffenheit der Einzelobjekte festgelegt: ein Objekt kann ein Punkt, eine Fläche oder ein dreidimensionales Objekt sein. Ein solches Einzelobjekt wird als ein Partikel (des Systems) bezeichnet.

„Gemein ist allen Ansätzen, daß Partikeln neben ihrer Grundform noch diverse andere Eigenschaften zugeordnet werden können. Diese Eigenschaften können je nach System und Einsatzgebiet variieren.“

(Hein 2002, 7)

Diese Eigenschaften bestimmen das Verhalten der Partikel. Dazu gehören Parameter wie die Lebenszeit und Geburtskoordinaten (Position des Emitters). Partikel werden also von einem oder mehreren Punkten aus emittiert und überleben eine bestimmte Dauer, bevor sie verschwinden.

Welche Bewegungen sie in ihrer Lebenszeit zurücklegen bestimmen die physikalischen Rahmenbedingungen. Hierbei können zwei unterschiedliche Prinzipien angewendet werden:

- geringe Komplexität: die Partikel bewegen sich mit einer Beschleunigung zu einem oder mehreren Punkten
- hohe Komplexität: den Partikeln wird eine Masse, und der Umwelt eine Gravitation zugeordnet. Die Partikel verhalten sich daraufhin frei nach den physikalischen Gesetzen (vgl Hein 2002, 7).

Zusätzlich zum Grad der Komplexität des gesamten Systems kann das visuelle Erscheinungsbild durch differierende Eigenschaften der Einzel- partikel entscheidend geprägt werden. Emittierende Partikel bekommen nicht gleiche Eigenschaften zugeschrieben, sondern bewegen sich in einem von GestalterIn/ProgrammiererIn gesteckten Rahmen. Die Masse eines Partikel hat in einem solchen Fall beispielsweise nicht den Wert 5, sondern einen zufälligen Wert zwischen 4 und 6.

Unabhängig von den gewählten Bedingungen lässt sich das Prinzip Partikelsystem nach William Reeves, dem Macher des Effekts aus Star Trek II, in folgende Schritte gliedern:

„To compute each frame in a motion sequence, the following sequence of steps is performed:

- (1) new particles are generated into the system,*
- (2) each new particle is assigned its individual attributes,*
- (3) any particles that have existed within the system past their prescribed lifetime are extinguished,*
- (4) the remaining particles are moved and transformed according to their dynamic attributes, and finally*
- (5) an image of the living particles is rendered in a frame buffer.“*

(Reeves 1983, 93)

3.3.3 Partikelsysteme in VVVV

Die bisher genannten Eigenschaften von Partikelsystemen beziehen sich auf 3d-Animation und Special Effects.

Bei generativer Gestaltung und vor allem bei Echtzeit-Anwendungen wie VVVV, werden meist Systeme mit geringerer Komplexität eingesetzt.

Der Begriff Partikelsystem wird bereits verwendet, wenn mehrere kleine Elemente mit einfachen Methoden „organisch“ animiert werden. Eine komplexe Physikumgebung und permanentes Emittieren, sowie eine begrenzte Lebensdauer, sind nicht zwingend notwendig.

Beispiele hierfür folgen in den kommenden Unterkapiteln.

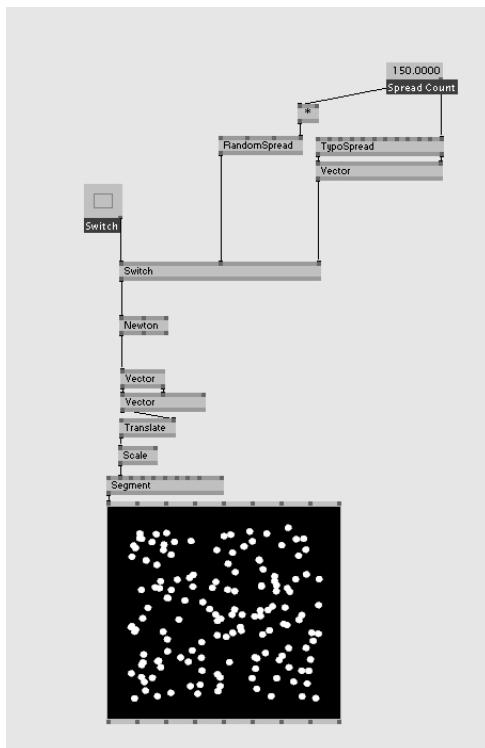
3.3.3.1 Partikelerzeugung mit Hilfe von Spreads

Das Verhalten vieler Punkte in VVVV kann unter anderem mit Hilfe der in Kapitel 1.3.2.4 vorgestellten Spreaderzeuger kontrolliert werden.

Die einfachste Anwendung stellt das Umschalten zwischen zwei Spreads dar, die für die Positionierung von Punkten verwendet werden.

Spread1 ist ein Randomspread, welcher zufällige Koordinaten erstellt – *Spread2* ist ein Typospread, der die Koordinaten auf der Outline eines Buchstabens ausgibt.

Ein Schalter (Switch) der zwischen die Werteausgänge der Spreads gesetzt wird, erlaubt ein hartes Hin- und Herschalten zwischen *Spread1* und *Spread2*.



img 3.3.02

Dieser Umschaltprozess kann mit Hilfe eines Newton Knotens (siehe *img 3.3.02* „abgefedert“ und somit durch dessen Interpolation¹ eine flüssige Bewegung erreicht werden.

Da es sich bei der Art der Interpolation um eine beschleunigte Bewegung handelt, wirkt dieses Umschalten der Koordinaten bereits wie eine physikalisches Phänomen und ist die einfachste Art ein Partikelsystem in VVVV zu erzeugen.

¹ Die Zwischenwerte werden berechnet

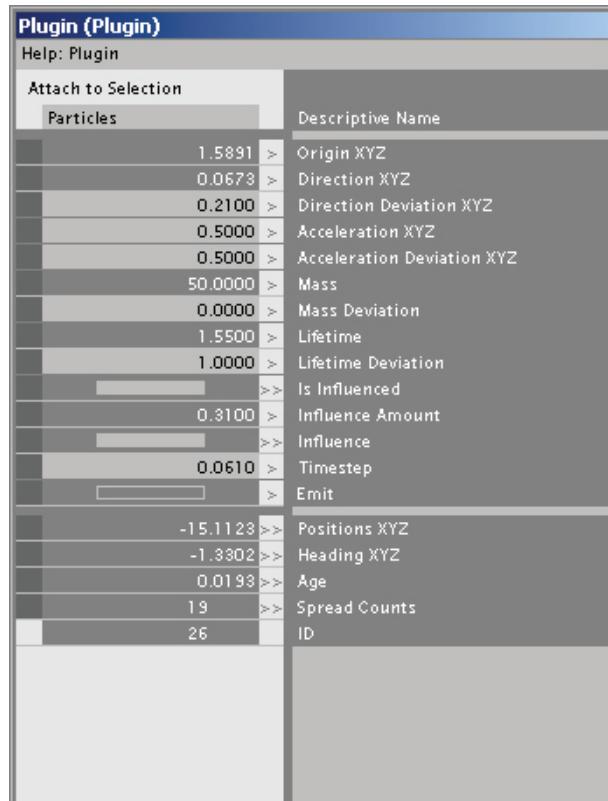
Folgende Abbildung *img 3.3.03* zeigt den Ausgangszustand *Spread1* (links), und den Endzustand *Spread2* (rechts). Die Stadien dazwischen sind interpolierte Zwischenergebnisse des Newton Knotens.



img 3.3.03

3.3.3.1 Partikel Plugin in VVVV

Für komplexe Partikelsysteme existiert in VVVV ein Plugin, das eine Vielzahl von Parametern für ein komplexen Verhalten von Massenpunkten bereitstellt.



img 3.3.04

Wie *img 3.3.04* zeigt, kann aufgrund der Parameter von einer physikalischen Simulation gesprochen werden:

Masse, Beschleunigung und Richtung wirken sich auf die Grundbewegung der Objekte aus, während die Lebenszeit den Sterbezeitpunkt festlegt. All diesen Werten kann eine Abweichung (Deviation)

zugewiesen werden, die Unregelmäßigkeiten wie in *Kapitel 3.2.2* erwähnt, hervorrufen kann.

Realistisches Schwarmverhalten kann zusätzlich durch gegenseitige Beeinflussung der Partikel erreicht werden.

Der Rechenaufwand dieser Operation ist jedoch immens, weshalb dieses Feature in Echtzeit Anwendungen bisher nicht oft Verwendung findet.

3.3.4 Typografische Anwendungen

Für typografische Betrachtung wird an dieser Stelle die Arbeit *Caligraft* (Ricard Marxer Piñón, 2006) der hinzugezogen.

Caligraft ist eine Sammlung von Processing Applikationen rund um generative Erzeugung und Veränderung von Typografie¹. Acht Anwendungen erlauben dem Benutzer mit der Maus Einfluss auf die Parameter von generativ erzeugten Schriften zu nehmen. In vielen der Applikationen werden Partikelsysteme verwendet.

Caligraft besitzt, aufgrund seines Forschungscharakters und seinen-gleichzeitig großen gestalterischen Qualitäten, Relevanz für das Forschungsgebiet.

3.3.4.1 Partikel in ihrer Reinform: Punkte

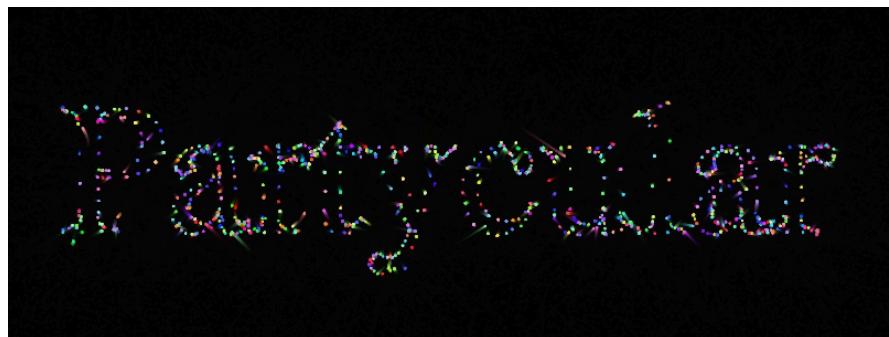
Naheliegend ist der Ansatz, Schriftzeichen aus Punkten zu gestalten und im Anschluss diese Punkte durch Bewegungsregeln zu Partikeln werden zu lassen.

Caligraft beinhaltet eine Applikation namens Pyrographie, die eine Vielzahl von zufällig positionierten Punkten generiert und diese an ihre Endposition – der Outline der Schrift – einpendeln lässt:

„Inspired by pyrotechnics, this calligraphy explores the limits to recognition. Particles of random colors travel around the screen searching for their position to form the characters.“
(Marxer Piñón 2006a)

Im Anschluss können BenutzerInnen mit der Maus interagieren und die Punkte von ihrer Endposition abbringen. Dabei werden nicht einzelne Punkte herumgeschubst sondern, ähnlich einem Windstoß, mehrere Punkte gleichzeitig bewegt. Je nach Intensität der Mausbewegung positioniert sich das visuelle Ergebnis zwischen lesbar und unlesbar (siehe *img 3.3.05 bis img 3.3.08*).

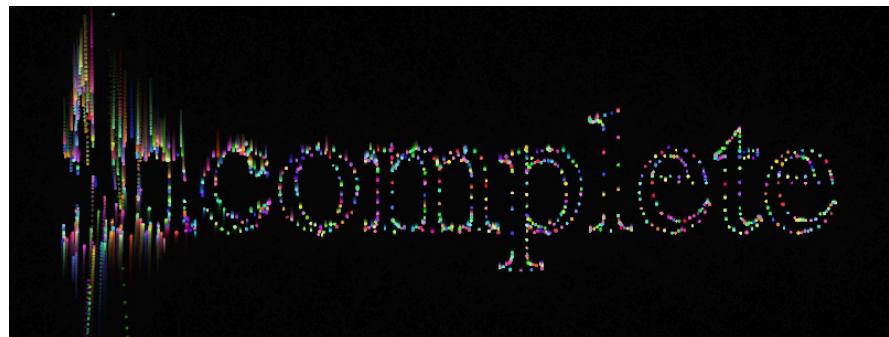
¹ Alle diese Anwendungen stehen unter www.caligraft.com unter Creative Commons Lizenz zur freien Verfü-gung



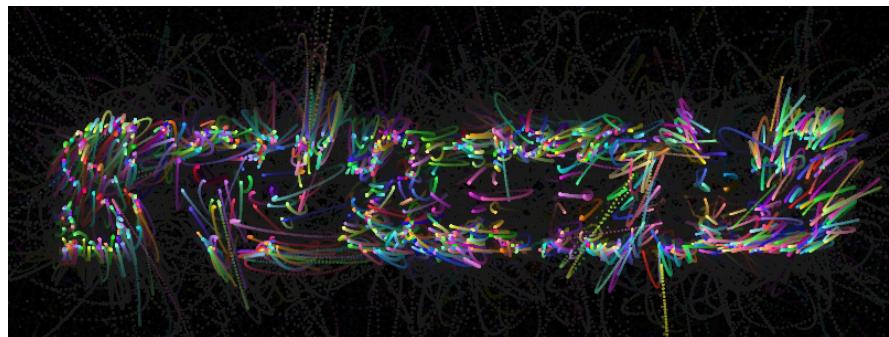
img 3.3.05



img 3.3.06



img 3.3.07



img 3.3.08

Die Generativität der Gestaltung funktioniert in diesem Fall sehr gut:
Der Benutzer wählt den Repräsentanten der Bildklasse mit der Maus

und kann so verschiedene, aber stets gestalterisch ansprechende, Ergebnisse produzieren.

Der Einsatz generativer Methoden ist hier, durch die Komplexität der Bewegung und der Einzelbilder, sichtlich gerechtfertigt.

Aus typografischer Sicht ist festzuhalten, dass bei dieser Anwendung die gewählte Schrift auf Punkte ihrer Outline reduziert wird. Darunter leidet die Lesbarkeit – der gestalterische Effekt rechtfertigt dies jedoch.

Weitere Darstellungsarten sind dennoch erstrebenswert.

3.3.4.2 Partikel kreieren Linien

Ein Ansatz der über die gepunktete Ästhetik hinausgeht, ist Partikel Linien zeichnen zu lassen.

Um einen Punkt zur Linie werden zu lassen, ist eine zusätzliche Dimension notwendig. Hierfür bietet sich die Zeit an – dh die Bewegung von Partikeln wird über eine bestimmte Zeit lang aufgezeichnet, so dass ein Partikel zur Linie wird.

Die Strategie für die Erstellung von Buchstaben muss somit eine andere sein: Statt wie im vorangegangenen Beispiel Punkte auf der Outline zu positionieren, müssen sich die Punkte (in diesem Fall würde auch ein einzelner Punkt reichen) an der Outline entlang bewegen. Dabei muss die Information des vergangenen Bewegungsablaufes gespeichert werden.

Dieser Fall wird in der Caligrafont Applikation Hilos behandelt:

„This piece is based on the idea of particles swimming along the character outlines. The parameters of the swim are controlled with the position of the mouse (...)“

(Marxer Piñón 2006b)

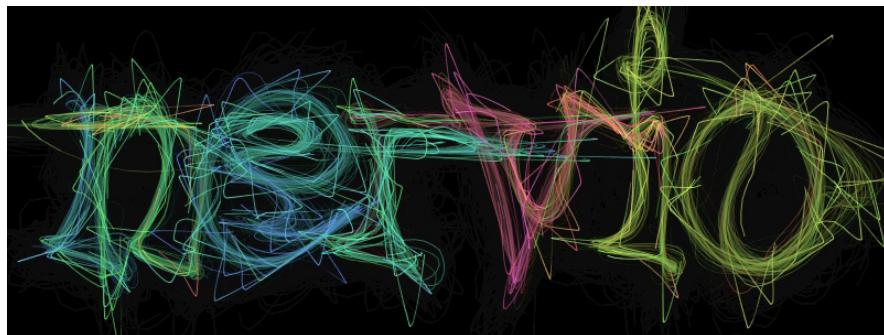
Mit den Schwimmparametern meint Marxer Piñón die Werte, die die Partikelbewegung auf der Outline bestimmen. Eine generative Bildklasse ergibt sich erst, wenn eine Variation dieser Bewegung ermöglicht ist. Diese kann erzeugt werden, indem die Partikel nicht direkt an der Outline entlang „schwimmen“, sondern durch zusätzliche Trägheit die Outline nicht erreichen oder darüberhinaus schießen (siehe *img*



img 3.3.09



img 3.3.10



img 3.3.11

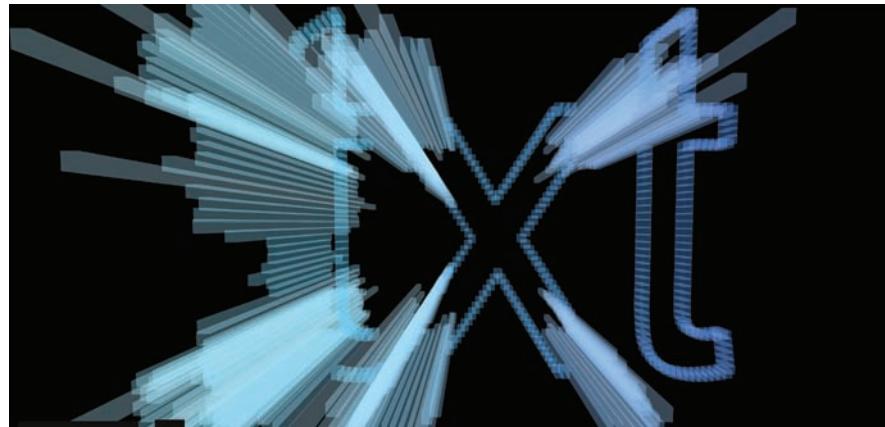
3.3.09 bis img 3.3.11).

Vorangestellte Beispiele liefern den Beweis, dass die Anwendung von Partikelsystemen geeignet ist, um generativ Displayschriften zu erzeugen.

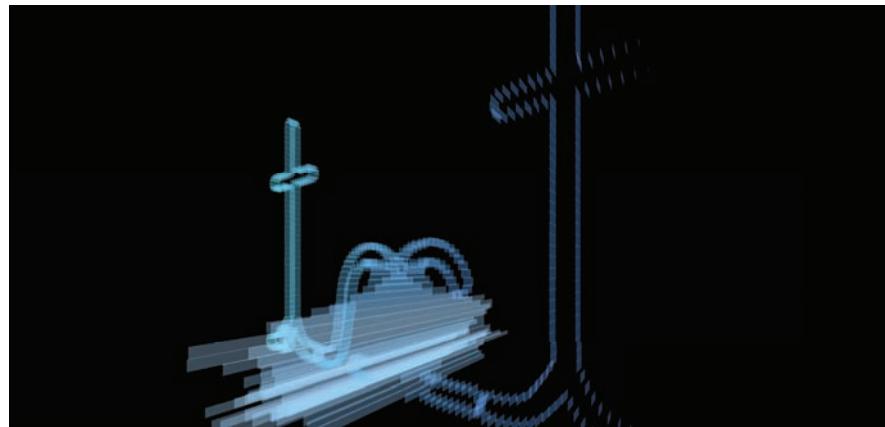
Die Generativität sowie das Prinzip der Bildklasse ist deutlich erkennbar.

Typografisch ist das Prinzip der Liniendarstellung das ausgeklügeltere von beiden, da die einzelnen Buchstaben zusammenhängender erscheinen, und somit mehr Lesbarkeit bzw Identifikation von Schrift gegeben ist. Dies führt zu einem größeren möglichen Einsatzgebiet.

Als Abschluss dieses Unterkapitels folgen einige Darstellungen des Autors, die sich der genannten Prinzipien bedienen oder diese weiterdenken.



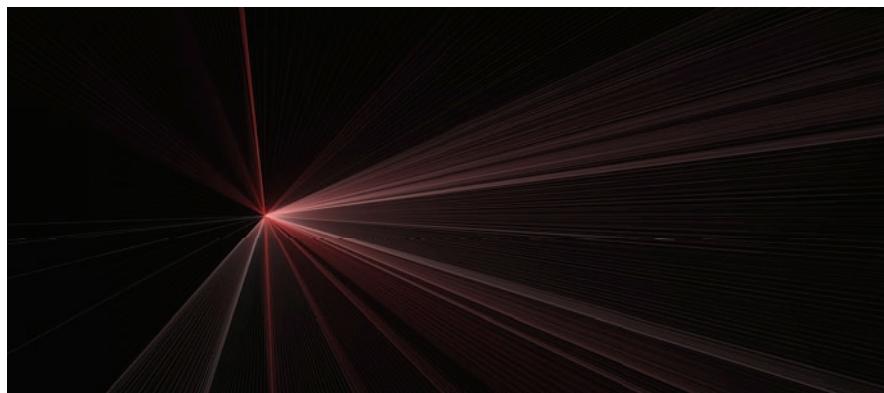
img 3.3.12



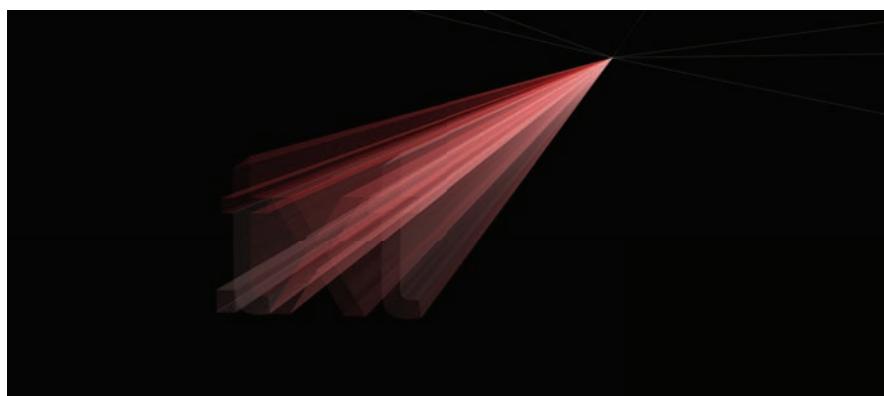
img 3.3.13

img 3.3.12 und *img 3.3.13* sind Auszüge aus dem der Arbeit beiliegenden Werk *touch.txt*. Es handelt sich um ein verfremdetes Partikelsystem: oragnische Bewegungen werden nicht auf die Position der Partikel, sondern auf deren Größe angewendet.

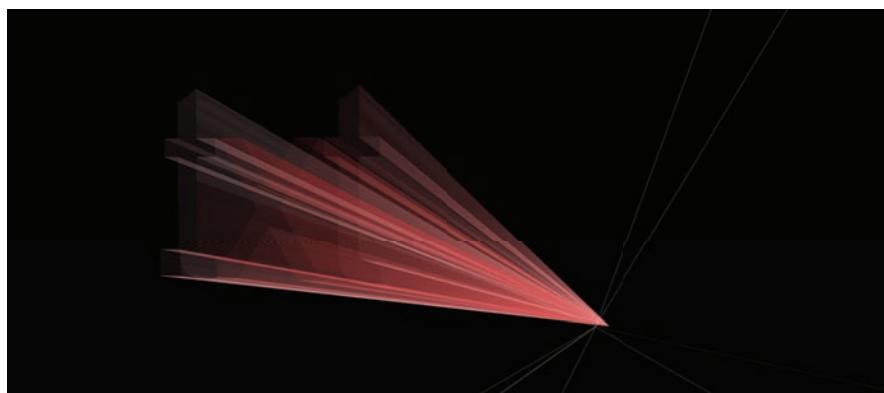
Je nach Wahl der Parameter funktioniert diese Anwendung auch gut mit detailreichen Serifen- und Schmuckschriften.



img 3.3.14



img 3.3.15



img 3.3.16

Bei img 3.3.14 - img 3.3.16 handelt es sich ebenfalls um Auszüge aus dem beiliegenden Werk *touch.txt*. Auch hier wird ein anderes Prinzip eines Partikelsystems versucht: Die Partikel haben die untypische Form von Linien, die von ihrem Startpunkt auf der Outline der Buchstaben, hin zu einem vom Benutzer gewählten Punkt gespannt werden. Lesbarkeit ist dadurch anfangs nicht gegeben, sondern muss vom Nutzer erst erarbeitet werden.

3.4. Attraktoren

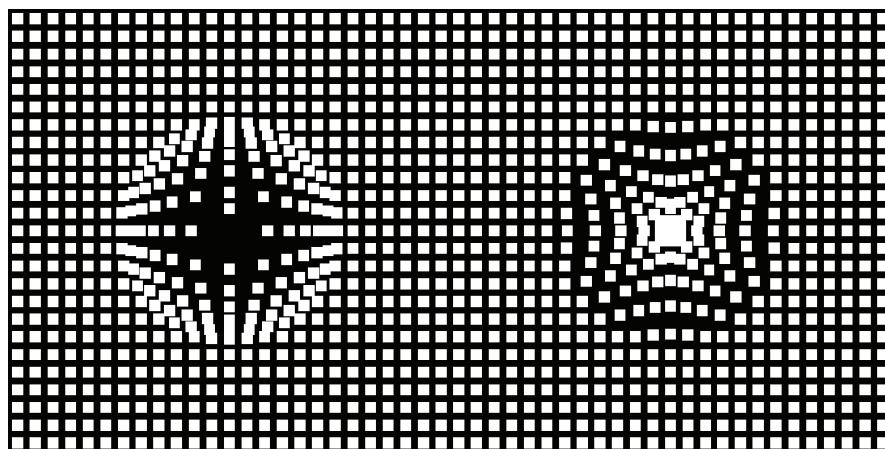
3.4.1 Attraktoren in VVVV

Attraktoren unterscheiden sich in VVVV vom Attraktorbegriff der Mathematik.

In VVVV handelt es sich um einen Anziehungspunkt, der Koordinaten in einem bestimmten Radius anzieht oder abstoßt. Der Unterschied ist, dass es sich beim mathematischen Attraktor um ein dynamisches System handelt, in dem die Koordinaten erst im Laufe der zeitlichen Entwicklung dem Einfluss des Attraktors folgen (vgl Meyers 2007). In VVVV verschiebt der Attraktor die Koordinaten sofort, und eine Dynamik ist ebenfalls keine Voraussetzung.

Die dynamischen und zeitlichen Phänomene sind in VVVV zwar implementierbar, jedoch nicht im Attraktorknoten integriert.

Die Abbildung *img 3.4.01* zeigt die Wirkungsweisen des *Attractor (2d)* Knotens auf die Anordnung von Quads: links werden die Koordinaten vom Punkt abgestoßen, rechts angezogen.

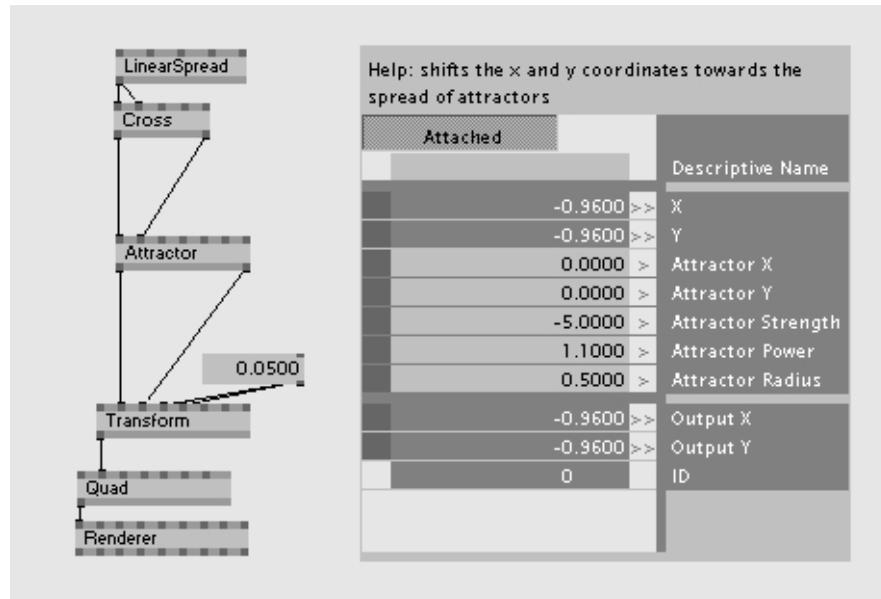


img 3.4.01

Der Inspektor aus *img 3.4.02* verdeutlicht die Handhabung: in die zwei Eingänge links werden die X und Y Koordinaten des Gesamtsystems eingespeist – im Fall von *img 3.4.01* das Raster auf dem sich die weissen Quads ursprünglich befinden. Die Pins daneben legen die Attraktorkoordinaten fest. Dieser Eingang kann auch mehrere Werte verarbeiten, was Beispiele im weiteren Verlauf des Kapitels aufzeigen.

Die restlichen drei Inputpins bestimmen das Attraktorverhalten:

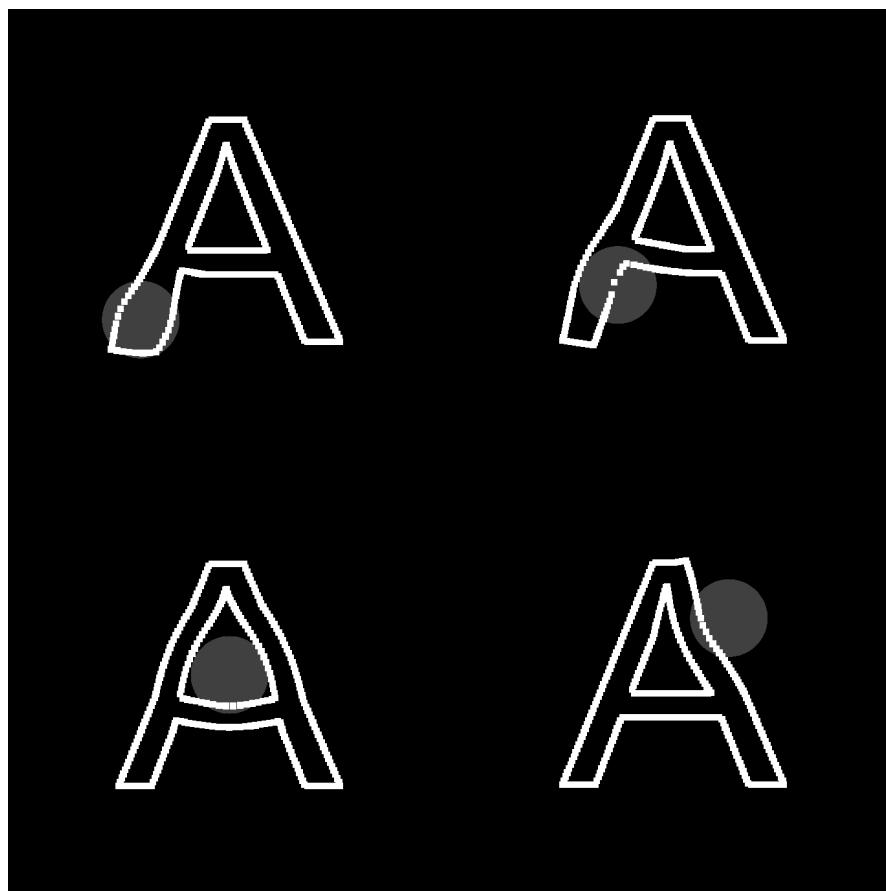
Zusätzlich zur Attraktorstärke und -Power, die entscheiden ob und wie stark die Koordinaten angezogen/abgestoßen werden, kann am *Attractor (2d)* Knoten der Attraktorradius festgelegt werden.



img 3.4.02

3.4.2 Typografische Anwendungen

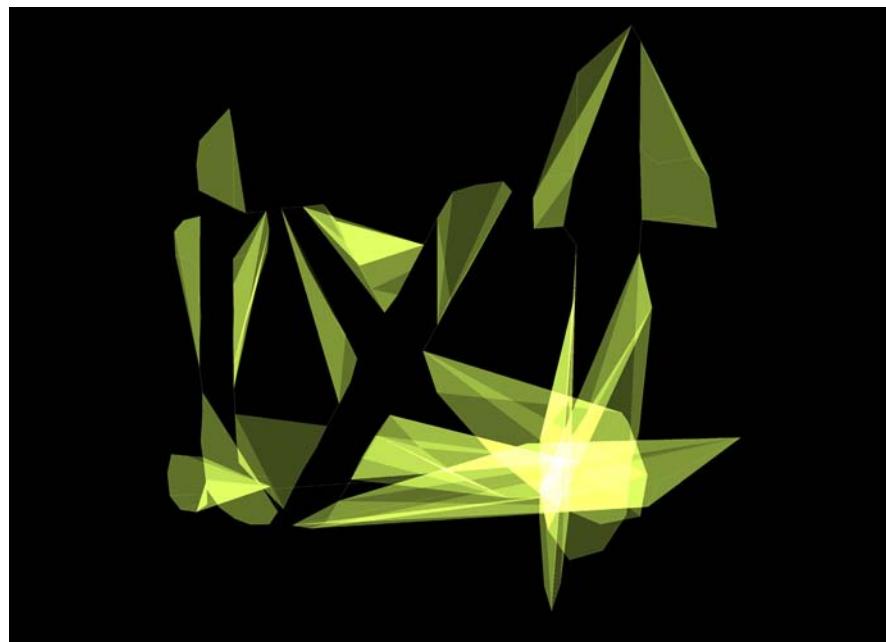
Für eine erste typografische Annäherung werden Koordinaten des TypoSpreads als Gesamtsystem in den Attraktor gegeben, an Stelle des Rasters im Beispiel oben. Der Attraktorpunkt ist dabei verschiebbar, was zur Folge hat dass sich der Buchstabe an den entsprechenden Stellen verformt (siehe *img 3.4.03*). Für besseres Verständnis wird der Attraktor von einem grauen Kreis repräsentiert.



img 3.4.03

Der Effekt ähnelt zunächst einem Verzerrungsfilter, wie er zB in Photoshop zu finden ist. Der entscheidende Unterschied ist, dass jedoch Koordinaten in ihrer Position verschoben werden, bevor das Bild gerendert wird. Photoshop hat hingegen lediglich die Möglichkeit das fertig gerendertes Bild zu verzerren.

Dieses denkbar einfache Prinzip kann in Kombination mit bereits vorgestellten Methoden und weiteren Techniken bereits visuell spektakuläre Effekte hervorbringen.



img 3.4.04

img 3.4.04 zeigt die Anwendung eines Attraktors auf das in *img 3.1.09* gezeigte Prinzip der zufälligen Flächenteilung. Der Attraktor wird in diesem Beispiel deutlich sichtbar durch die Überlagerungen der halbtransparenten Flächen, die zu einem hellen Punkt in der Komposition führen.

Typografisch ist diese Anwendung oft problematisch, da die Lesbarkeit nur zu erahnen ist. Mit Serifenlosen Schriften funktioniert dies jedoch im Rahmen von Displayschriften ausreichend. Bei detailreichen Serifen- oder Schmuckschriften müssen die Parameter schon sehr genau gewählt werden, um noch von Schrift sprechen zu können.

img 3.3.04 ist ein Auszug aus dem beiliegenden Werk *touch.txt*. In der Werkdokumentation können weitere Repräsentanten der Bildklasse betrachtet werden.



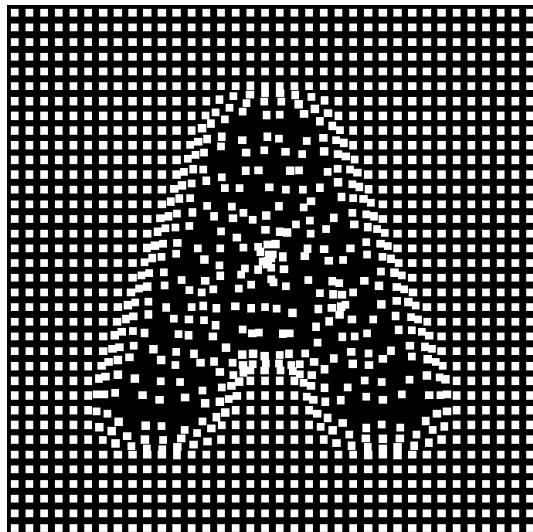
img 3.4.05

Bei obiger Abbildung (*img 3.4.05*) handelt es sich ebenfalls um einen Auszug aus dem der Arbeit zugehörigem Werk.

Hierbei wird ein Partikelsystem mit Attraktoren kombiniert. Die von Attraktoren beeinflusste Pseudophysik des Partikelsystems führt zu sehr organisch anmutenden Ergebnissen. Durch die so implementierte Zeitabhängigkeit nähern sich die Attraktoren in dieser Anwendung dem Attraktorbegriff der Mathematik an.

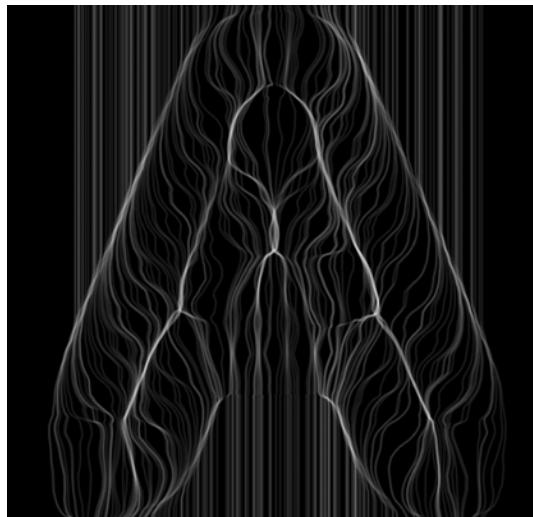
Bei den beiden Beispielen wird deutlich, dass Attraktoren eine geeignete Möglichkeit sind, um generativ erzeugte mit zusätzlichen Parametern Schriften zu beeinflussen. Das grundsätzliche, visuelle Erscheinungsbild kann dabei variieren und auf weiteren Methoden generativer Gestaltung basieren.

Im folgenden wird das obige Prinzip der Attraktoranwendung umgekehrt: statt Schrift als Basis des Systems zu verwenden, wird sie nun für die Positionen von Attraktoren eingesetzt (siehe *img 3.4.06*).



img 3.4.06

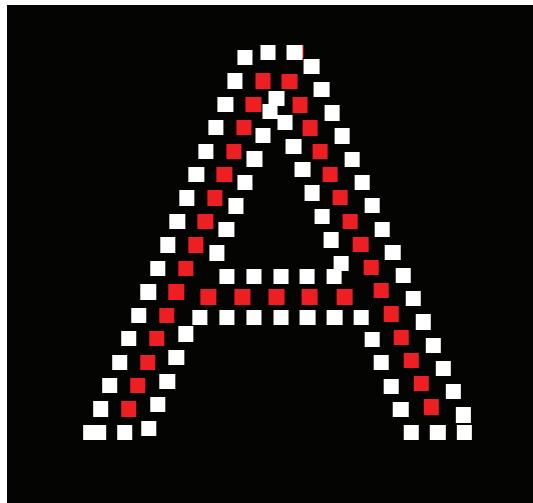
Dieses Beispiel zeigt bereits auf, dass der Buchstabe nur erahnt werden kann. Die Parametriesierung gestaltet sich schwierig, vor allem wenn mehrere Buchstaben dargestellt werden sollen.



img 3.4.07

Mit anderen Geometrien wie zum Beispiel Linien kann zwar ein ästhetischeres Ergebnis erreicht werden, jedoch zeigt sich auch erstmals der Typospread von VVVV als ungeeignet: da sich die Attraktoren auf der Outline der Schrift platzieren (entsprechend den weissen Rechtecken

in *img 3.4.08*), wird jede Linie durch zwei Attraktoren beschrieben. Geeigneter wäre ein Knoten, der die Schrift auf eine Linie reduziert, und Koordinaten ausgibt die den roten Rechtecken in *img 3.4.08* entsprechen.



img 3.4.08

Ein solches Verfahren würde mehr Lesbarkeit gewähren, und ein breites Anwendungsspektrum für diesen besonderen Attraktoreinsatz ermöglichen.

Auch wenn der zweite in diesem Kapitel vorgestellte Ansatz keine wirklich zufriedenstellenden Ergebnisse hervorbringt, sind Attraktoren als sehr geeignet für generative Displayschriften einzustufen. Ihre Rolle ist jedoch mehr der Beeinflussung anderer Methoden zugeordnet. Eine Schrift nur aus Attraktoren und Primitiven zu erstellen ist bringt keine ausreichenden visuellen Ergebnisse hervor.

Im, der Arbeit beiliegenden Werk werden Attraktoren häufig eingesetzt, da sich ihre Parameter gut auf die dort verwendete Technik übertragen lassen, und somit eine funktionierende Benutzerinteraktion mit der Bildklasse zustande kommt.

4. **Schluss**

Erklärtes Ziel dieser Arbeit ist es, die Tauglichkeit spezifischer Methoden generativer Gestaltung in VVVV, hinsichtlich der Erzeugung von Displayschriften zu untersuchen.

Dabei wurde der Forschungsfrage nachgegangen, ob generative Methoden generell dafür geeignet sind, und wenn ja, welche Methoden sich besonders dafür anbieten.

Für die Untersuchung ausschlaggebend ist zu allererst die Begriffsdefinition der generativen Gestaltung, die im ersten Kapitel hergeleitet wird.

Das zweite Kapitel grenzt den Begriff Typografie ein auf *Erzeugung von Displayschriften* ein, und gibt Aufschluss über den Prozess der Erzeugung digitaler Typografie am Bildschirm. Dies ist eine Vorbereitung, Prozessschritte zu finden, an denen generativ eingegriffen werden kann.

Im anschließenden Kapitel werden diese Eingriffe, unter Berücksichtigung spezifischer Methoden generativer Gestaltung, vorgenommen.

Die vom Autor für diese Arbeit gewählten Methoden haben sich, nach einigen gedanklichen Schritten, alle als geeignet herausgestellt Displayschriften zu generieren. Dabei wurde stets untersucht, ob ein gestalterischer Mehrwert gegeben ist, und typografisch relevante Eigenschaften aufgezeigt. Zusätzlich wurde der Begriff und der Einsatz der generativen Gestaltung hinterfragt.

Trotz dieser Kriterien, ist das Ergebnis jeder Untersuchung positiv ausgefallen. Ein solch einseitiges Ergebnis bietet eine Diskussionsgrundlage über die Relevanz der Untersuchungen.

Ist es sinnvoll spezifische Methoden auszuwählen, und wäre eine allgemeine Antwort nicht hilfreicher?

Die einzelnen Abschnitte des dritten Kapitels zeigen, dass jede Methode ihre Strategie verlangt, generative Displayschriften hervorzubringen. Es existiert kein Patentrezept um aus der Methode Zufall eine Schrift zu erzeugen. Um mit Hilfe des Zufalls eine Schrift zu formen, können viele Herangehensweisen gewählt werden, die allesamt parameterbasiertes Gestalten voraussetzen. Die Herangehensweisen die Partikelsysteme erfordern, unterscheiden sich davon gänzlich. Ziel der Arbeit ist es, solche Herangehensweisen und den damit verbundenen Gestaltungsprozess zu verdeutlichen.

Da die Methoden so spezifisch sind, ist es nicht möglich diese Einblicke allgemein zu formulieren.

Ein weiterer Diskussionspunkt ist die Schriftklasse, die der Autor für seine Untersuchungen gewählt hat. Was rechtfertigt Untersuchungen an Displayschriften, wenn Lesbarkeit dabei kaum ein Kriterium ist? Wäre es dann nicht dasselbe, von einem geometrischen Grundobjekten auszugehen?

Methoden generativer Gestaltung können von geometrischen Primitiven ausgehen und daran ausreichend erläutert werden. Jedoch grenzt die Verwendung von Schrift das Forschungsfeld ein, auch wenn es sich um Displayschrift handelt. Dies ist notwendig um zusätzliche Kriterien bei der Wertung hinzuziehen zu können. Die typografischen Kriterien sind in vielen Punkten ausschlaggebend, die Strategie noch einen Schritt weiter zu denken, oder in eine andere Richtung zu lenken. Dies wäre bei der Gestaltung mit Primitiven willkürlich.

Um sich dem Diskussionspunkt von der anderen Seite zu nähern muss hinterfragt werden, wieso nicht eine andere Schriftklasse gewählt wurde. Diese Entscheidung wurde vom Autor im zweiten Kapitel bereits begründet: Textschriften oder andere Schriftklassen, die mehr Funktion erfordern, sind für den Rahmen dieser Arbeit, mit diesen Methoden zu komplex. Auf dieser Arbeit können diesbezüglich Forschungen aufgebaut werden.

Darüberhinaus fällt auf, dass alle untersuchten Methoden positiv abgeschnitten haben. Dabei stellt sich die Frage, ob nur Methoden ausgewählt wurden die positive Ergebnisse hervorbringen – oder ob sich alle Methoden generativer Gestaltung eignen, Displayschriften zu erzeugen.

Nach dieser Forschungsarbeit kann in der Tat darauf geschlossen werden, dass alle Methoden generativer Gestaltung geeignet sind. Jedoch mit der Einschränkung, dass jede Methode ihre spezifische Strategie mit sinnvolle Parametrisierung verlangt. Dies bedeutet unterschiedlich hohen Aufwand.

In Kapitel 3.4.2 wird die Möglichkeit gezeigt, Attraktoren auf der Outline von Schrift zu positionieren. Diese Anwendung wurde zunächst als wenig geeignet eingestuft, jedoch existiert gleichzeitig der Ausblick, dass mit aufwändigeren Methoden sinnvolle Anwendungen zustande kommen können.

Mit der Parametrisierung verhält es sich ähnlich. Wie *img 3.2.02* aufzeigt, kann die Methode Zufall sinnvoll angewandt werden – falsche oder übertriebene Parameter führen jedoch zu Ergebnissen, die nichts mehr mit Schriftgestaltung gemein haben.

Die generativen Gestaltungsmethoden an sich, die Algorithmen, sind stets in der Lage auf gestalterisch ansprechende Weise Werte zu generieren oder zu verändern, wenn sie ausreichend durchdacht angewendet werden. Eine grundsätzliche Einschränkung für typografische Anwendung ergibt sich, unter oben genannten Voraussetzungen, nicht.

Das dieser Arbeit zugehörige Werk rundet die Forschungen ab.

Die interaktive Installation lässt den Benutzer verschiedene Kombinationen von Gestaltungsmethoden hinsichtlich der Bildklasse erfahren. Der Autor gibt so die Parameter aus der Hand, und lässt den Benutzer zwischen lesbar und unlesbar, zwischen Displayschrift und Chaos alleine. Es wird lediglich eine grobe Eingrenzung vorgenommen, um die Suche nach den Parametern der Lesbarkeit nicht zu schwer zu gestalten.

Technische Details können dem Anhang „Werkdokumentation“ entnommen werden.

Als ein weiterer Forschungsbereich bietet sich die generative Erzeugung von Textschriften an. Dabei muss der Mehrwert ein anderer, sowie die visuellen Ergebnisse subtiler sein, als in der vorliegenden Arbeit der Fall. Interessant wäre hierbei, ob Generativität der Textschrift der Lesbarkeit dienen kann.

Es könnte durch generative Eingriffe zum Beispiel kommuniziert werden, wie lange das gerade gelesene Kapitel noch ist, um so ein neugieriges Vorblättern überflüssig zu machen.

Der Ansatz Informationsgehalt in generative Typografie zu packen, eröffnet gleichzeitig ein eigenes Forschungsgebiet. Eine solche Anwendung kann zum Beispiel bei Wegleitsystemen passieren: statt „WC 50m rechts“ könnten die Buchstaben des Wortes „WC“ generativ so beeinflusst werden, dass der Benutzer intuitiv weiß, dass er in 50 Metern rechts gehen muss.

Schrift kann so zusätzlicher Informationsträger sein, und quantitativ mehr Inhalt kommunizieren, als das der Sinnzusammenhang ihrer Buchstaben vollbringen mag.

Generative Typografie wird in ihren Platz in der Gestaltung finden – ob BetrachterInnen sie als solche identifizieren können, ist eine andere Frage.

Vorallem interaktive Installationen und Bewegtbilddesign bieten sich als Anwendungsgebiete an, da die Bildklasse so durch Animation, oder benutzergestaltete Parameter begreifbar wird. Dieses Phänomen ist in der Lage die BetrachterInnen in Staunen zu versetzen, da die daraus resultierende Formensprache noch nicht ansatzweise ausgeschöpft ist.

„The computer solves everything except the essential. The essential is still reserved for men.“

(Kelemen zit n Nierhoff-Wielk 2007, 53)

Literaturverzeichnis

Bücher, Sammelbände

BILZ 2004

BILZ, Silja (2004): *Type-One. Discipline and Progress in Typography*.
Berlin: Die Gestalten.

COUFFIGNALS 1955

COUFFIGNALS, Louis (1955): *Denkmaschinen*.
Stuttgart: Kilpper.

FRANKE 1957

FRANKE, Herbert W. (1957): *Kunst und Konstruktion*.
München: Bruckmann.

FRANKE 2007

FRANKE, Herbert W. (2007): *Experimentelle Ästhetik (Interview)*.
In: ERZOGENRATH, Wulf / NIERHOFF-WIELK, Barbara (Hg): *Ex Machina – Frühe Computergrafik bis 1979*.
Berlin: Deutscher Kunstverlag 108-133.

FRY/REAS 2007

FRY, Ben / REAS, Casey (2007): *Processing. A Programming Handbook for Visual Designers and Artists*.
Cambridge (USA): The MIT Press.

HÖRISCH 2004

HÖRISCH, Jochen (2004): *Eine Geschichte der Medien. Von der Oblate zum Internet*.
Frankfurt am Main: Suhrkamp.

KLÜTSCH 2007

KLÜTSCH Christoph / LANFERMAN Petra / NIERHOFF-WIELK Barbara (2007):
Stationen der frühen Computerkunst.
In: HERZOGENRATH, Wulf / NIERHOFF-WIELK, Barbara (Hg): *Ex Machina – Frühe Computergrafik bis 1979*.
Berlin: Deutscher Kunstverlag 230-292.

MAEDA 1999

MAEDA, John (1999): *Design by Numbers*.
Cambridge (USA): The MIT Press.

NEES 1969

NEES, Georg (1969): *Generative Computergrafik*.
Berlin: Siemens.

NIERHOFF-WIELK 2007

NIERHOFF-WIELK, Barbara (2007): *Die Begegnung von Computer und Kunst. Ein Blick zurück*.
In: HERZOGENRATH, Wulf / NIERHOFF-WIELK, Barbara (Hg): *Ex Machina – Frühe Computergrafik bis 1979*.
Berlin: Deutscher Kunstverlag 20-63.

PIEHLER 2007

PIEHLER, Heike (2007): *Herbert W. Franke und die Entdeckung neuer Bilderswelten*.
In: HERZOGENRATH, Wulf / NIERHOFF-WIELK, Barbara (Hg): *Ex Machina – Frühe Computergrafik bis 1979*.
Berlin: Deutscher Kunstverlag 64-107.

ROSEN/WEIBEL 2007

ROSEN, Margit / WEIBEL, Peter (2007): *Die Zukunft eines künstlerischen Mediums*.
In: HERZOGENRATH, Wulf / NIERHOFF-WIELK, Barbara (Hg): *Ex Machina – Frühe Computergrafik bis 1979*.
Berlin: Deutscher Kunstverlag 182-229.

TROGEMANN/VIEHOFF 2005

TROGEMANN, Georg / VIEHOFF, Jochen (2005): *Code@Art. Eine elementare Einführung in die Programmierung als künstlerische Praxis*.
Wien: Springer.

WILLBERG 2003

WILLBERG, Hans Peter (2003): *Wegweiser Schrift. Erste Hilfe für den Umgang mit Schriften*
Mainz: Hermann Schmidt.

Artikel, wissenschaftliche Arbeiten

BRÜLL/SCHWARZER 2006

BRÜLL, David / SCHWARZER, Björn (2006): Graphical Web Server Programming. Recherche, Konzeption und prototypische Implementierung einer web-basierten Unterstützung graphischer Programmierung auf Basis von VVVV. Diplomarbeit, Fachhochschule/Darmstadt.

DER SPIEGEL 1965

Der Spiegel (1965): Bald krumme Linien.
In: Der Spiegel 18/1965. Hamburg: Spiegel Verlag 151-152.

HEIN 2002

HEIN, Johannes (2002): Partikelsysteme.
Hausarbeit, Universität/Ulm.
In: <http://medien.informatik.uni-ulm.de/lehre/courses/sso2/Computergrafik/JohannesHein.pdf> – aufgerufen am 08.09.2008, 19 Uhr.

HEINSCH 2007

HEINSCH, Ingolf (2007): Entwurf und Implementierung einer zeitleistenbasierten Parametersteuerung in eine datenstromorientierte Echtzeitprogrammierumgebung.
Diplomarbeit, Fachhochschule für Technik und Wirtschaft/Berlin.

HITTHALER 2005

HITTHALER, Thomas (2005): Generative Erzeugung von Design mit VVVV.
Diplomarbeit, Fachhochschule/Salzburg.

MANFÉ/MAIRITSCH 2008

MANFÉ, Michael / MAIRITSCH, Karin (2008): Diplom – Leitfaden 2008.
Salzburg: FH Salzburg.

MALE 2005

MALE, Christian (2005): Methodik der Apparative Kunst.
Diplomarbeit, Fachhochschule/Salzburg.

REEVES 1983

REEVES, William (1983):
Particle Systems. A Technique for Modeling a Class of Fuzzy Objects.
In: ACM Transactions on Graphics, Vol. 2, No. 2, April 1983, 91-108.

SCHIRN KUNSTHALLE 2007a

Schirn Kunsthalle (2007): Maschinenkunst – Kunstmaschinen. Presseinformation.

Frankfurt am Main: Schirn Kunsthalle.

SCHIRN KUNSTHALLE 2007b

Schirn Kunsthalle (2007): OpArt. Presseinformation.

Frankfurt am Main: Schirn Kunsthalle.

SUTHERLAND 1963

SUTHERLAND, Ivan E. (1963): Sketchpad. A Man-machine Graphical Communication System.

In: Proceedings – Spring Joint Computer Conference. Detroit (USA): Joint Computer Conference 507-524.

Onlinequellen

BEINERT 2008a

BEINERT, Wolfgang (2008): Typographie. Definition, Etymologie und Geschichte.

In: <http://typolexikon.de/t/typographie.html> – aufgerufen am 14.08.2008, 14 Uhr

BEINERT 2008b

BEINERT, Wolfgang (2008): Westeuropäische Schriftgeschichte.

Vom Kerbzeichen zur digitalen Schriftlichkeit

In: <http://typolexikon.de/s/schriftgeschichte.html> – aufgerufen am 14.08.2008, 14 Uhr

BEYARS 2008a

BEYARS Kunstlexikon (2008): Das große Kunstlexikon von P.W. Hartmann. Konstruktivismus.

In: http://www.beyars.com/kunstlexikon/lexikon_4989.html – aufgerufen am 31.05.2008, 14 Uhr.

BEYARS 2008b

BEYARS Kunstlexikon (2008): Das große Kunstlexikon von P.W. Hartmann. Neoplastizismus.

In: BeyArs Kunstlexikon: http://www.beyars.com/kunstlexikon/lexikon_6300.html – aufgerufen am 30.04.2008, 11 Uhr.

GROPIUS 1919

GROPIUS, Walter (1919): Bauhaus Manifest.

In: <http://www.bauhaus.de/bauhaus1919/index.htm> – aufgerufen am 14.08.2008, 12 Uhr.

MARXER PIÑÓN 2006a

MARXER PIÑÓN, Ricard (2006): Pyrographie.

In: <http://www.caligraft.com/exhibition/pyrographie> – aufgerufen am 11.08.2008, 13 Uhr.

MARXER PIÑÓN 2006b

MARXER PIÑÓN, Ricard (2006): Hilos.

In: <http://www.caligraft.com/exhibition/hilos> – aufgerufen am 11.08.2008, 13 Uhr.

MEYERS 2007

Meyers Lexikon Online 2.0 (2007): Attraktor.

In: <http://lexikon.meyers.de/meyers/Attraktor> – aufgerufen am 29.08.2008, 12 Uhr

MICROSOFT 1997

Microsoft Corporation (1997): TrueType Fundamentals.

In: <http://www.microsoft.com/typography/otspec/TTCH01.htm> – aufgerufen am 14.08.2008, 13 Uhr.

MICROSOFT 1997b

Microsoft Corporation (1997): TrueType fonts.

In: <http://www.microsoft.com/typography/TrueTypeFonts.mspx> – aufgerufen am 06.06.2008, 13 Uhr.

MICROSOFT 2001

Microsoft Corporation (2001): OpenType Overview.

In: <http://www.microsoft.com/typography/otspec/otover.htm> – aufgerufen am 14.08.2008, 13 Uhr.

MICROSOFT 2008

Microsoft Developer Network (2008): Windows GDI Start Page.

In: [http://msdn.microsoft.com/en-us/library/ms536795\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms536795(VS.85).aspx) – aufgerufen am 14.08.2008, 13 Uhr.

RADIOHEAD 2008a:

RADIOHEAD (2008): RA DIOHEA_D / HOU SE OF_C ARDS

In: <http://code.google.com/creative/radiohead/> – aufgerufen am 14.08.2008, 12 Uhr.

SCHREIBER 2000

SCHREIBER, Alfred (2000): Was ist ein Algorithmus?

In: http://www.gefilde.de/ashome/vorlesungen/algorithmen/algo_abschn11/algo_abschn11.html – abgerufen am 09.03.2008, 14 Uhr.

VVVV.org 2007:

VVVV (2007): Executive FAQ.

In: <http://vvvv.org/tiki-index.php?page=Executive+FAQ> – aufgerufen am 06.08.2008, 14 Uhr.

VVVV.ORG 2008a

VVVV (2008): VVVV – a multipurpose toolkit.

In: <http://vvvv.org> – aufgerufen am 01.06.2008, 14 Uhr.

VVVV.ORG 2008b

VVVV (2008): VVVV – Propaganda.

In: <http://vvvv.org/tiki-index.php?page=propaganda> – aufgerufen am 02.06.2008, 11 Uhr.

VVVV.ORG 2008c

VVVV (2008): VVVV – RandomSpread (Spreads).

In: [http://vvvv.org/tiki-index.php?page=RandomSpread+\(Spreads\)](http://vvvv.org/tiki-index.php?page=RandomSpread+(Spreads)) – aufgerufen am 14.08.08, 15 Uhr.

WEISSTEIN 2008

Weisstein, Eric W. (2008): Rectangle.

In: Mathworld: <http://mathworld.wolfram.com/Rectangle.html> – aufgerufen am 30.04.2008, 11 Uhr.

WIKIPEDIA 2008a

Wikipedia (2008): Wikipedia. Gestaltung.
In: <http://de.wikipedia.org/wiki/Gestaltung> – aufgerufen am
31.05.2008, 13 Uhr.

WIKIPEDIA 2008b

Wikipedia (2008): Lissajous-Figur.
In: <http://de.wikipedia.org/wiki/Lissajous-Figur> – aufgerufen am
06.09.2008, 15 Uhr.

WIKIPEDIA 2008c

Wikipedia (2008): Randomness,
In: <http://en.wikipedia.org/wiki/Randomness> – aufgerufen am 07.09.2008,
18 Uhr.

WRIGTH 2006

ENO, Brian / WRIGHT, Will (2006): Fora.tv. Playing with time.
In: http://fora.tv/fora/fora_transcript_pdf.php?cid=355 – aufgerufen am
29.05.2008, 12 Uhr.

Alle Artikel, wissenschaftlichen Arbeiten und Onlinequellen befinden sich im Datenverzeichnis der Werk DVD.

Abbildungsverzeichnis

- img 1.2.01*, Seite 20 Tanya Bonakdar Gallery (2005): Olafur Eliasson, The endless study.
In: <http://www.artnet.de/magazine/reviews/pschak/pschak09-14-06.asp>
- img 1.2.02*, Seite 20 Tanya Bonakdar Gallery (2005): Olafur Eliasson, The endless study.
In: <http://www.artnet.de/magazine/reviews/pschak/pschak09-14-06.asp>
- img 1.2.03*, Seite 23 Herbert W. Franke (1953-55): Lichtformen.
In: http://www.dam.org/franke/franke53-73_0068.html
- img 1.2.04*, Seite 23 Herbert W. Franke (1953-55): Lichtformen.
In: http://www.dam.org/franke/franke53-73_0068.html
- img 1.2.05*, Seite 24 Herbert W. Franke (1956): Oscillogramm.
In: http://www.dam.org/franke/franke53-73_0058.html
- img 1.2.06*, Seite 26 A. Michael Noll (1965): Computer Compositions with Lines.
In: <http://pratt.edu/~llaurola/cg550/cg.htm>
- img 1.2.07*, Seite 26 Piet Mondrian (1917): Compositions with Lines.
In: <http://pratt.edu/~llaurola/cg550/cg.htm>
- img 1.3.01*, Seite 31 Philipp Steinweber (2008): Scriptographer Interface.
- img 1.3.02*, Seite 32 Philipp Steinweber (2008): Scriptographer stich.js.
- img 1.3.03*, Seite 33 Philipp Steinweber (2008): Scriptographer Ergebnisse.
- img 1.3.04*, Seite 35 Philipp Steinweber (2008): VVVV Summe.
- img 1.3.05*, Seite 36 VVVV Group (2006): DataTypes.
In: <http://www.org/tiki-index.php?page=screenshots>
- img 1.3.06*, Seite 36 Philipp Steinweber (2008): VVVV Startup.
- img 1.3.07*, Seite 37 Philipp Steinweber (2008): VVVV Nodemenü.
- img 1.3.08*, Seite 38 Philipp Steinweber (2008): VVVV komplexer Patch.
- img 1.3.09*, Seite 39 Philipp Steinweber (2008): VVVV Quad.
- img 1.3.10*, Seite 40 Philipp Steinweber (2008): VVVV Spreads Übersicht.
- img 1.3.11*, Seite 41 Philipp Steinweber (2008): VVVV Spreads doof.
- img 1.3.12*, Seite 41 Philipp Steinweber (2008): VVVV Spredas klug.
- img 1.3.13*, Seite 42 Philipp Steinweber (2008): VVVV Spreads komplex.
- img 2.3.01*, Seite 57 Philipp Steinweber (2008): VVVV Text (GDI).
- img 2.3.02*, Seite 58 Philipp Steinweber (2008): VVVV TypoSpread.

- img 2.3.03, Seite 59* Philipp Steinweber (2008): Teapot.
- img 2.3.04, Seite 60* Philipp Steinweber (2008): VVVV 3d Text Wireframe.
- img 2.3.05, Seite 61* Philipp Steinweber (2008): VVVV Teapot Shading.
- img 2.3.06, Seite 62* Philipp Steinweber (2008): VVVV 3d Text.
- img 3.2.01, Seite 69* Philipp Steinweber (2008): Zufall Quads.
- img 3.2.02, Seite 72* Philipp Steinweber (2008): Zufall XY.
- img 3.2.03, Seite 73* Philipp Steinweber (2008): Zufall Z.
- img 3.2.04, Seite 74* Philipp Steinweber (2008): Zufälliges A.
- img 3.2.05, Seite 74* Philipp Steinweber (2008): A wird konkret.
- img 3.2.06, Seite 75* Philipp Steinweber (2008): Zufalls Mesh.
- img 3.2.07, Seite 76* Philipp Steinweber (2008): Zufalls Flächen.
- img 3.2.08, Seite 77* Philipp Steinweber (2008): Zufalls Mesh-Band.
- img 3.3.01, Seite 80* Paramount Pictures (1982): Star Trek II – The Wrath of Khan: Initial Explosion.
In: <http://www.double.co.nz/dust/Reeves-1983-PSA.pdf>
- img 3.3.02, Seite 82* Philipp Steinweber (2008): VVVV Particles simple.
- img 3.3.03, Seite 83* Philipp Steinweber (2008): VVVV Particles Animation.
- img 3.3.04, Seite 83* Philipp Steinweber (2008): Particle Plugin Inspektor.
- img 3.3.05, Seite 86* Ricard Marxer Piñón (2006): Partycular.
In: <http://www.caligraft.com/exhibition/pyrographie>
- img 3.3.06, Seite 86* Ricard Marxer Piñón (2006): Efface.
In: <http://www.caligraft.com/exhibition/pyrographie>
- img 3.3.07, Seite 86* Ricard Marxer Piñón (2006): Incomplete.
In: <http://www.caligraft.com/exhibition/pyrographie>
- img 3.3.08, Seite 86* Ricard Marxer Piñón (2006): Swarms.
In: <http://www.caligraft.com/exhibition/pyrographie>
- img 3.3.09, Seite 88* Ricard Marxer Piñón (2006): Chalk.
In: <http://www.caligraft.com/exhibition/hilos>
- img 3.3.10, Seite 88* Ricard Marxer Piñón (2006): Minim.
In: <http://www.caligraft.com/exhibition/hilos>
- img 3.3.11, Seite 88* Ricard Marxer Piñón (2006): Nervio.
In: <http://www.caligraft.com/exhibition/hilos>
- img 3.3.12, Seite 89* Philipp Steinweber (2008): Particle Scaling 1.
- img 3.3.13, Seite 89* Philipp Steinweber (2008): Particle Scaling 2.
- img 3.3.14, Seite 90* Philipp Steinweber (2008): Particle Beams 1.

- img 3.3.15, Seite 90* Philipp Steinweber (2008): Particle Beams 2.
- img 3.3.16, Seite 90* Philipp Steinweber (2008): Particle Beams 3.
- img 3.4.01, Seite 92* Philipp Steinweber (2008): Attractor in/out.
- img 3.4.02, Seite 93* Philipp Steinweber (2008): Attractor Inspektor.
- img 3.4.03, Seite 94* Philipp Steinweber (2008): Attractor moving.
- img 3.4.04, Seite 95* Philipp Steinweber (2008): Delaunay Attractor.
- img 3.4.05, Seite 96* Philipp Steinweber (2008): Heading Attractor.
- img 3.4.06, Seite 97* Philipp Steinweber (2008): Attractor Positions A.
- img 3.4.07, Seite 97* Philipp Steinweber (2008): Attractor Lines A.
- img 3.4.08, Seite 98* Philipp Steinweber (2008): Better Positions.

Glossar

Font

Schriftart (zB Arial, Helvetica, Garamond)

Fontdatei

Datei in der Schriftarten am Computer transportiert werden.

Gemeine/Minuskeln

Kleinbuchstaben

Glyph

ein einzelner Buchstabe

Serifen

„Füßchen“ einer Schrift (wie zB bei der Times New Roman)

Spationierung

individuelle Erweiterung der Laufweite eines Wortes

Versalien

Großbuchstaben

Anhang

Werkdokumentation

Beim Werk *touch.txt* handelt es sich eine interaktive Installation.

An einem Multitouch-Tisch können BenutzerInnen das Prinzip der Bildklasse an verschiedenen Anwendungen generativer Displayschriften erfahren.

Das Prinzip Multitouch ermöglicht überaus komplexe Interaktion. Mit einer Vielzahl von Fingern kann so Einfluss auf Gestaltung genommen werden. Merkwürdigerweise bestehen prominente Anwendungen, wie man sie aus Internetvideos vom iPhone kennt, primär aus Photobrowsern. Interaktion an Multitouchgeräten scheint sich primär auf die Zoom-Fingerbewegung zu beschränken, woraufhin die Motivation des Autors begründet liegt, Methoden zu finden die über dieses Prinzip hinausgehen.

In *touch.txt* werden also generative Gestaltungsmethoden für die Erzeugung von Displayschriften hin untersucht, und zusätzlich hinsichtlich ihrer Interaktionsmöglichkeiten präzisiert.

Auf der beiliegenden DVD befinden sich einerseits die VVVV Patches der Installation, die bei installiertem VVVV gestartet werden können. Die Interaktion ist aufgrund der fehlenden Multitouch-Umgebung nur sehr eingeschränkt mit der Maus möglich. Dies dient grobem Verständnis, ist aber lediglich ein Zusatz zur eigentlichen Installation.

Die Installation selbst wird in Form einer Videodokumentation, die sich ebenfalls auf der DVD befindet, veranschaulicht.

Für die Entwicklung der Installation *touch.txt* stellte das Designstudio Strukt¹ dem Autor den selbst entwickelten Multitouch-Tisch Prototyp zur Verfügung. Ähnlich der frühen Computerkunst wurde das Werk somit im „industriellen“ Forschungsumfeld erstellt.

Für das Werk wurde die dem Tisch zugehörige Trackingumgebung von Strukt benutzt. Das bedeutet, VVVV erhält vom Tisch die Koordinaten der Finger auf dem Tisch, sowie deren Druckstärke. Diese Wertelisten werden über eine Netzwerkverbindung in den *touch.txt* Patch gesendet. Die Gestaltung und Programmierung der sichtbaren Applikation stammt somit vom Autor.

¹ www.strukt.com

Die Hardware besteht aus dem Tisch, sowie einer Kamera und einem Projektor. Die Tischplatte wird von unten gefilmt, das Videosignal live in Koordinaten der Finger verarbeitet und gleichzeitig auf die Anwendung angewandt und projiziert.

Die für das Tracking verwendete Software ist ebenfalls VVVV.

Die einzelnen Applikationen werden in ein, vom Autor erstelltes, Framework geladen, um Basisfunktionen jederzeit zu gewährleisten.

Folgende sichtbare Interfaceelemente und Funktionen stehen dem Benutzer, unabhängig von der geladenen Applikation, somit immer zur Verfügung:

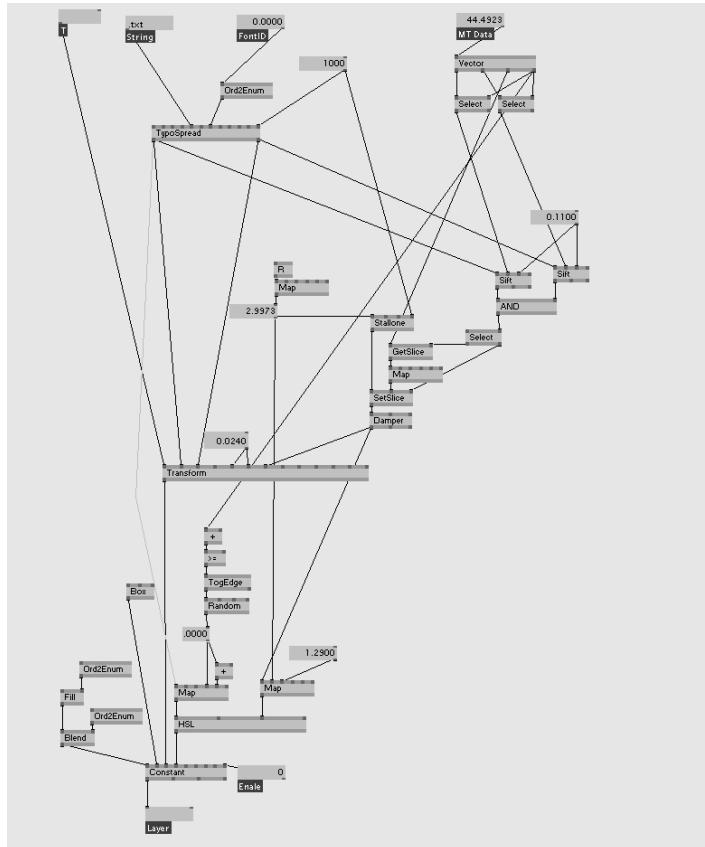
- Steuerung der Kamera: Drehen in alle Achsen, sowie Bewegen auf der z-Achse
- Auswählen von drei Schriftklassen: Serifenschrift (Georgia), seriflose Schrift (Arial), Schmuck-/Schreibschrift (Mamãe Que Nos Faz)
- Ein zusätzlicher, globaler Slider am unteren Tischrand welcher, je nach Applikation, eine andere Funktion zugeschrieben bekommt

Alle weiteren Interaktionsmöglichkeiten sind nicht explizit durch Interfaceelemente gekennzeichnet, sondern finden durch Berührung der Tischoberfläche statt.

Es folgt eine Auflistung und Erläuterung der einzelnen Applikationen.

Particle Boxes

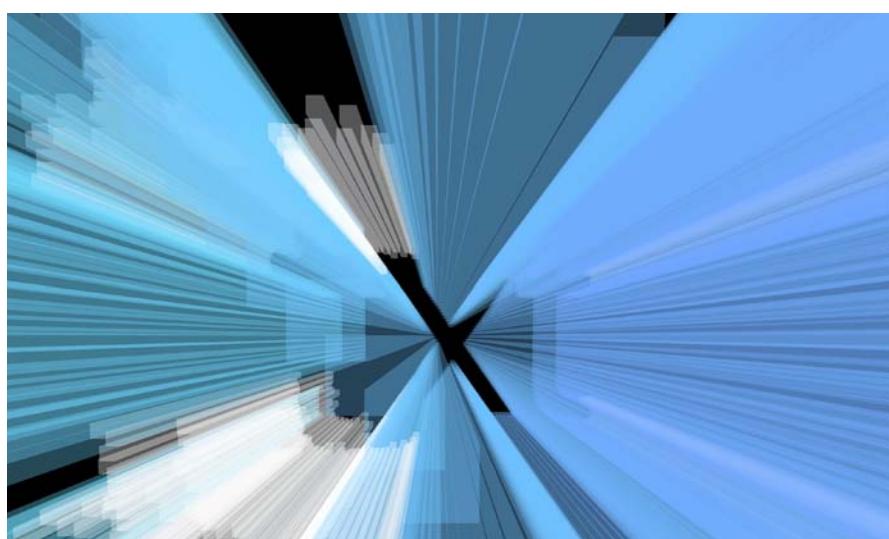
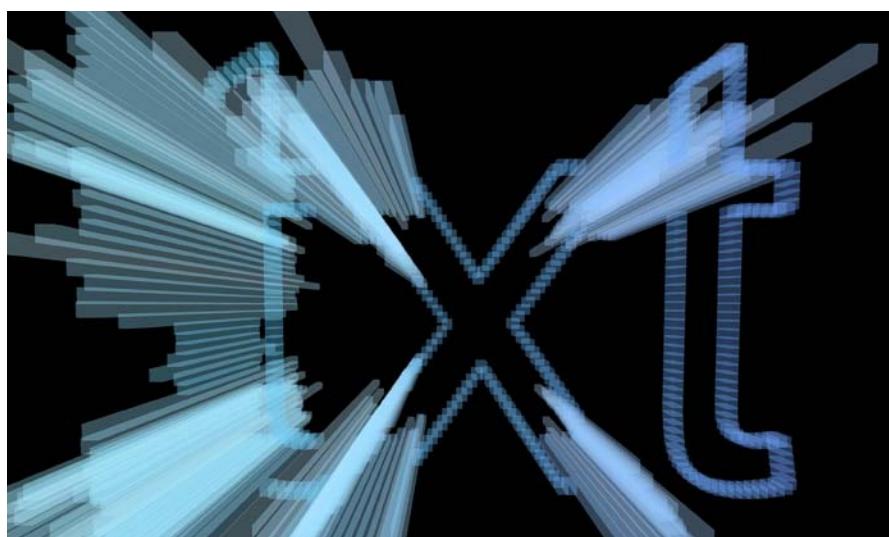
Particle Boxes ist ein geeignetes Beispiel, die Beschreibung der einzelnen Applikationen zu beginnen, da es technisch sehr überschaubar, und das Multitouch Prinzip äußerst nachvollziehbar angewendet ist.



Beim Grundsetting handelt es sich um Quader (*Box (EX9.Geometry)*), die mit Hilfe des Typospreads auf der Outline der Schrift positioniert werden.

Wird die Schrift von BenutzerInnen am Screen/Tisch berührt, skalieren sich diese Boxen, je nach Druckstärke in der z-Achse auf. Dabei greift das Prinzip des Multitouchscreens: es kann eine Vielzahl von Fingern gleichzeitig registriert werden, dh mehrere Leute können mit mehreren Fingern diesen Effekt auslösen.

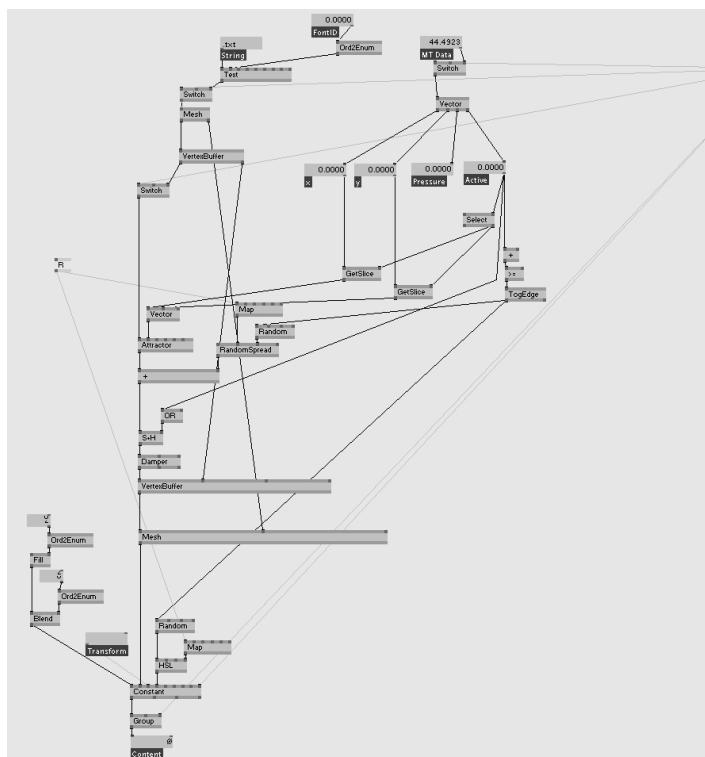
Der zusätzliche globale Slider erlaubt eine Umkehrung des Ausgangszustandes: wird er nach rechts geschoben, werden die Boxen von stark in der z-Achse skaliert und werden bei anschließender Berührung kleiner.



Random Mesh

Random Mesh nutzt den 3d Text von VVVV, um ihn mit Hilfe des Zufalls und Attraktoren zu verformen.

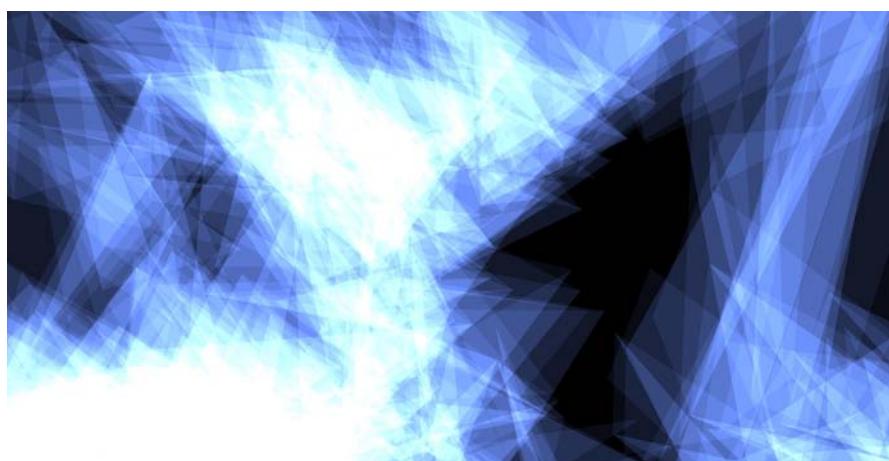
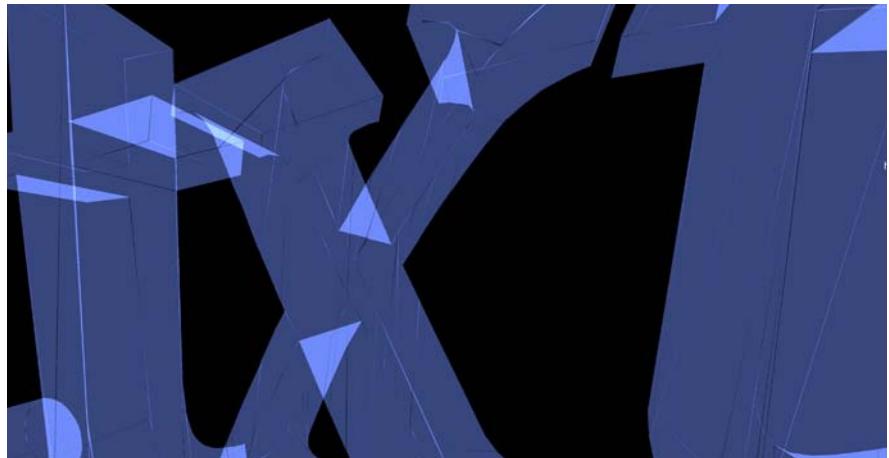
Dazu ist es notwendig, anfangs die Koordinaten des Polygonnetzes zu extrahieren, und in Form von Werten zugänglich zu machen. Diese Koordinaten dienen anschließend als Basissystem für den Attraktor.



Jeder Finger der den Tisch berührt, stellt im folgenden einen Attraktor dar; die Druckstärke wird auf die Attraktorenstärke angewendet.

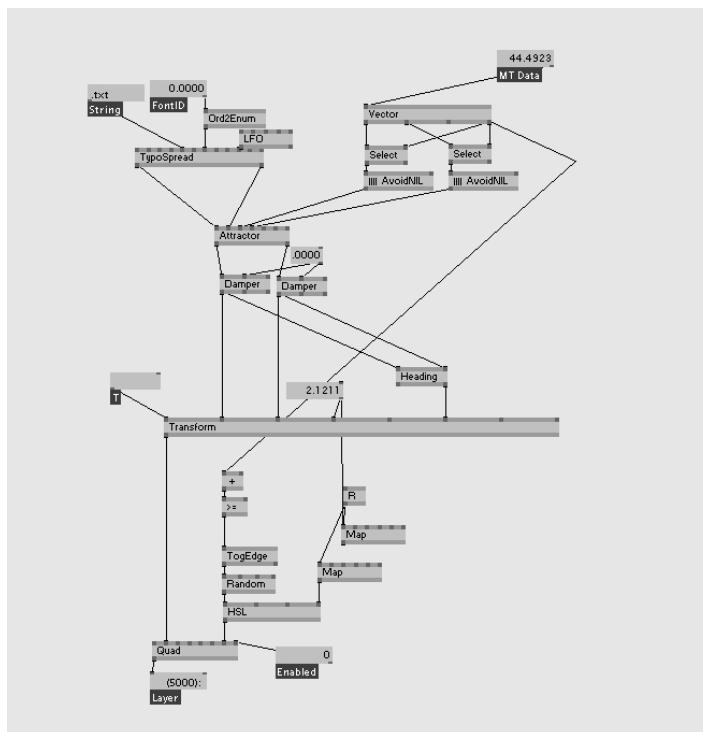
Die Attraktoren stoßen die Koordinaten ab, das hat zur Folge, dass die Schrift den Fingern auszuweichen scheint.

Mit Hilfe des globalen Sliders ist es möglich, Zufallswerte auf die Polygonnetz-Koordinaten zu addieren. Befindet sich der Slider ganz links, wird kein Zufall hinzu addiert – im Zustand ganz rechts ist sind die Werte hingegen so groß, dass Lesbarkeit nicht mehr gewährleistet ist.



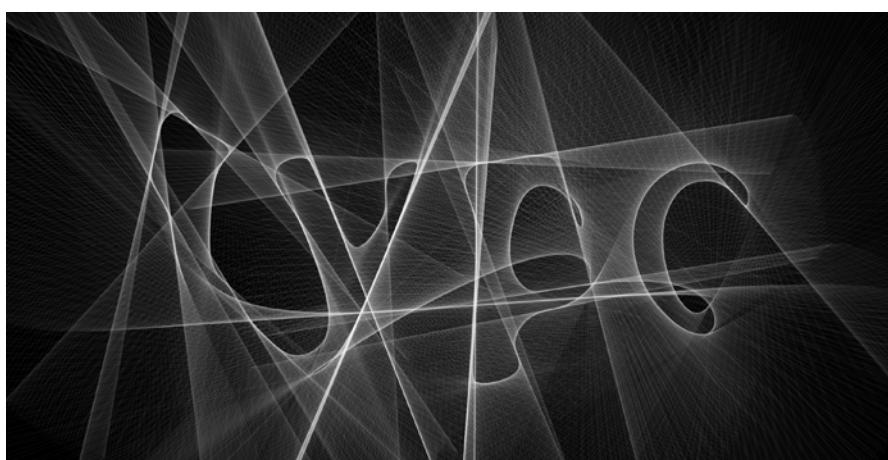
Heading Particles

Heading Particles ist ein geeignetes Beispiel aufzuzeigen, dass visuelle Komplexität auch durch einen kleinen Patch zustande kommen kann. Technisch gesehen werden erneut auf der Outline der Schrift Quads platziert: mit einer sehr kleinen Breite und einer einstellbaren Länge. Diese Quads „schwimmen“, wie in Kapitel 3.3.4.1 erläutert, an der Outline der Schrift entlang. Der *Heading (Animation)* Knoten sorgt für die besondere Drehung der Quads: jedes Quad dreht sich senkrecht zu seinem momentanen Richtungsvektor.



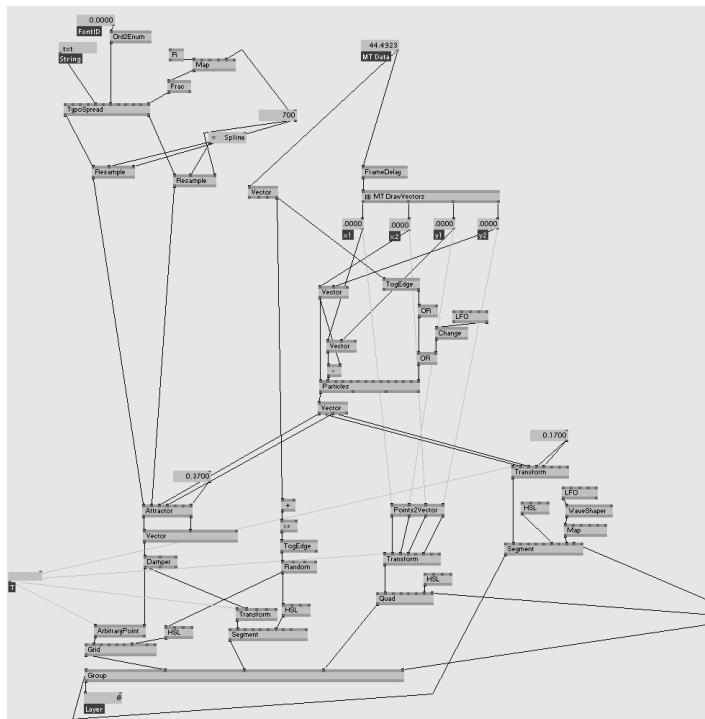
Der globale Slider erlaubt dem Benutzer die Länge der Quads festzulegen.

Die weitere Interaktion besteht aus Attraktoren, die, mit einer starken Trägheit versetzt, die Partikel anziehen. Dies führt zu einer rauchähnlichen Bewegungsästhetik, und erinnert an Frankes Oscillographien.



Attractor Cannon

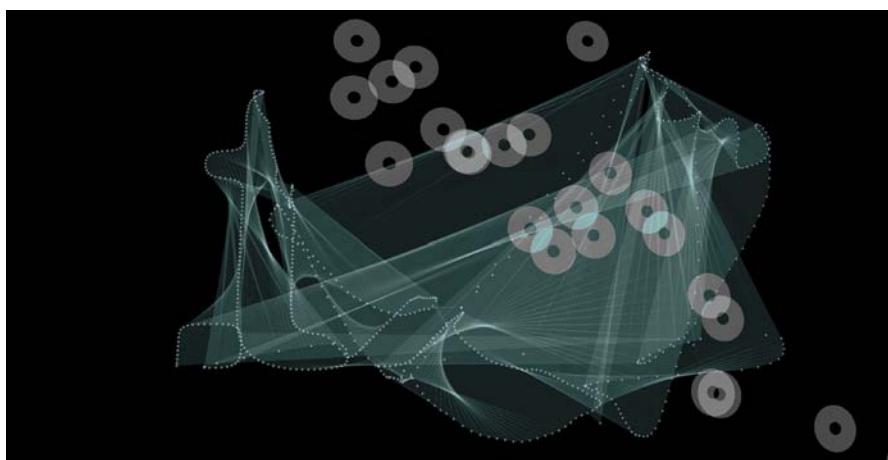
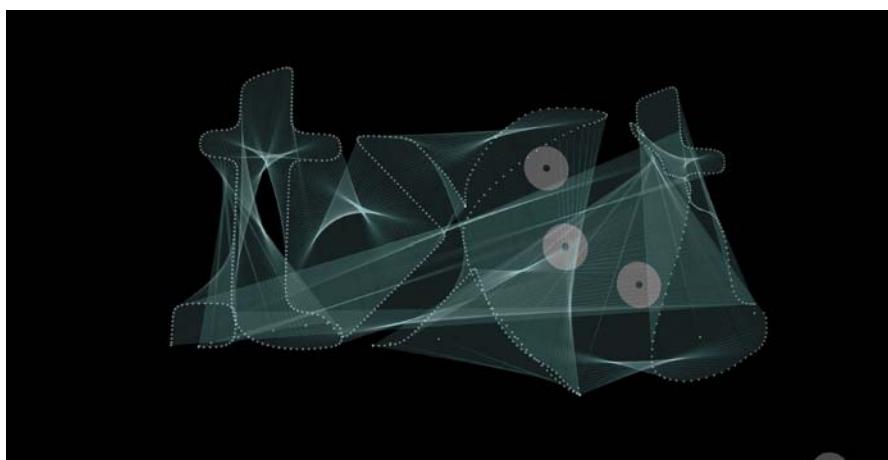
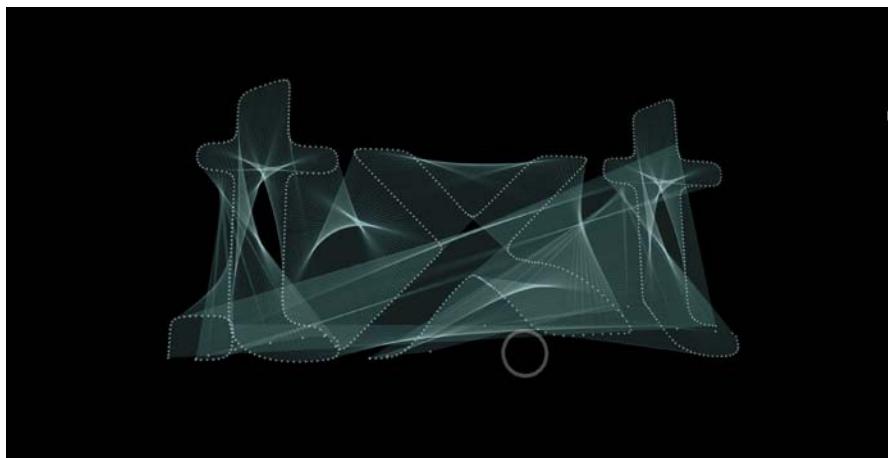
Die technische Basis der Attractor Cannon ist ein verformtes *Grid (DX9)*¹, welches sich Netzartig um die Schrift spannt.



Die BenutzerInnen können in dieser Anwendung Partikel schießen, die gleichzeitig Attraktoren sind. Mit einer eigens entwickelten Multitouch-Vektorenlogik kann mit jedem Finger eine Linie gezeichnet werden, die als Abschussvektor für die Partikel gilt. Wenn der entsprechende Finger auf dem Tisch gehalten wird, lässt sich so Richtung und Geschwindigkeit der Partikel ändern.

Im Bezug zur Schrift haben die Partikel eine abstoßende Wirkung, die sich auf das komplette Grid auswirkt, welches die Schrift umspannt.

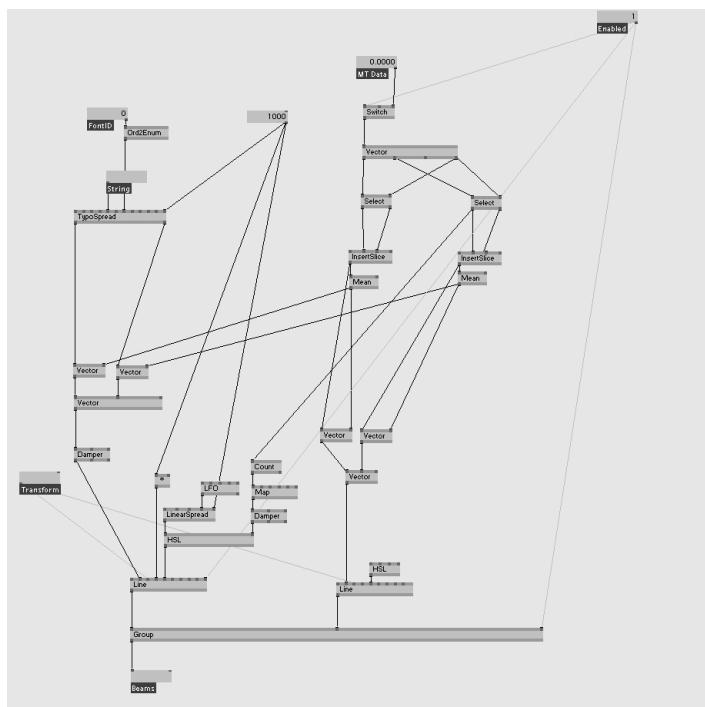
¹ Eine VVVV Geometrie, die in ihrer Grundform aussieht wie ein Quad. Jedoch ist sie mit speziellen Transformationen verformbar.



Particle Beams

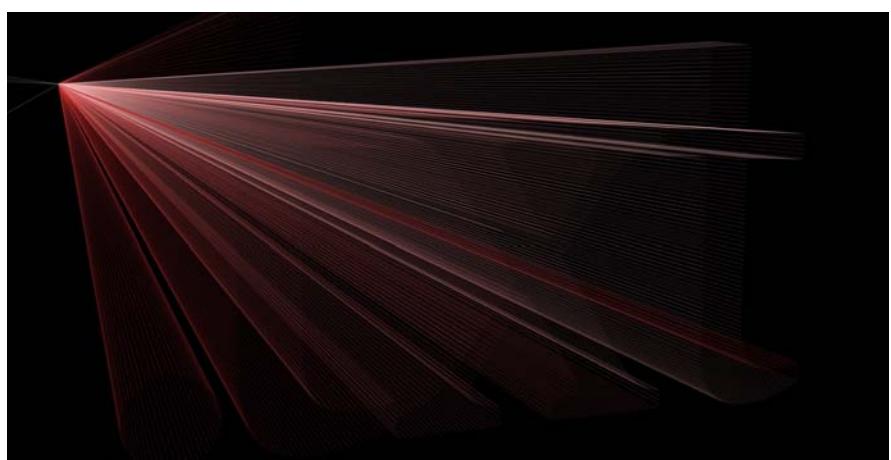
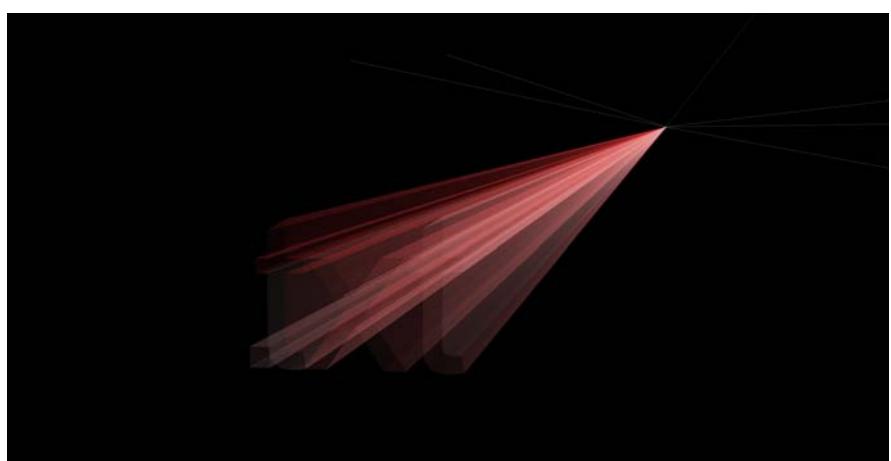
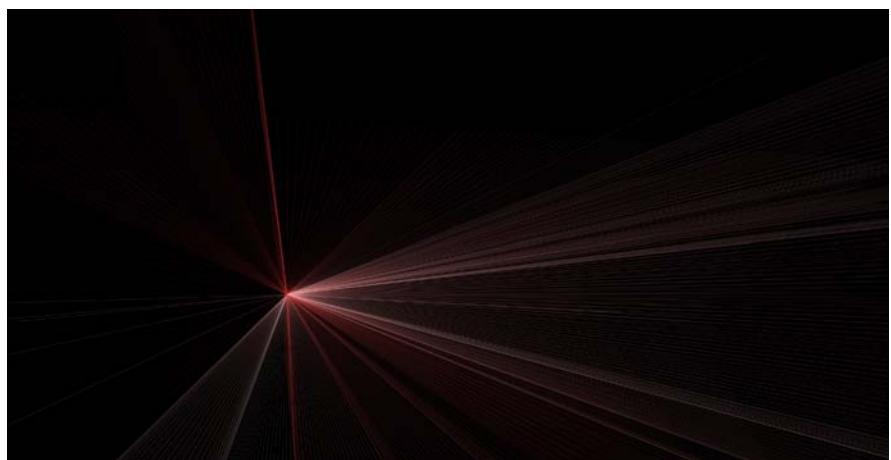
Auf die Particle Beams wird in Kapitel 3.3. bereits technisch eingegangen.

Die Partikel sind in diesem Fall Linien, die von der Schriftoutline aus alle in einem Punkt zusammenlaufen. Es existiert nur ein Parameter, der aber von mehreren Fingern der BenutzerInnen festgelegt wird. Dies erfolgt mit Hilfe der Bildung eines Mittelwerts aus allen x- und aus allen y-Koordinaten. Bei einem Finger bewegt sich der Zielpunkt mit dem Finger mit, sobald es sich um mehrere Finger handelt, befindet er sich genau in der mathematischen Mitte davon.



Zusätzlich wird durch die Anzahl der aktiven Finger die Transparenz der Linien beeinflusst. Je mehr Finger sich auf dem Tisch befinden, desto sichtbarer ist der Schriftzug. Generell kann eine bessere Lesbarkeit erreicht werden, wenn der Zielpunkt ausserhalb der Schrift liegt.

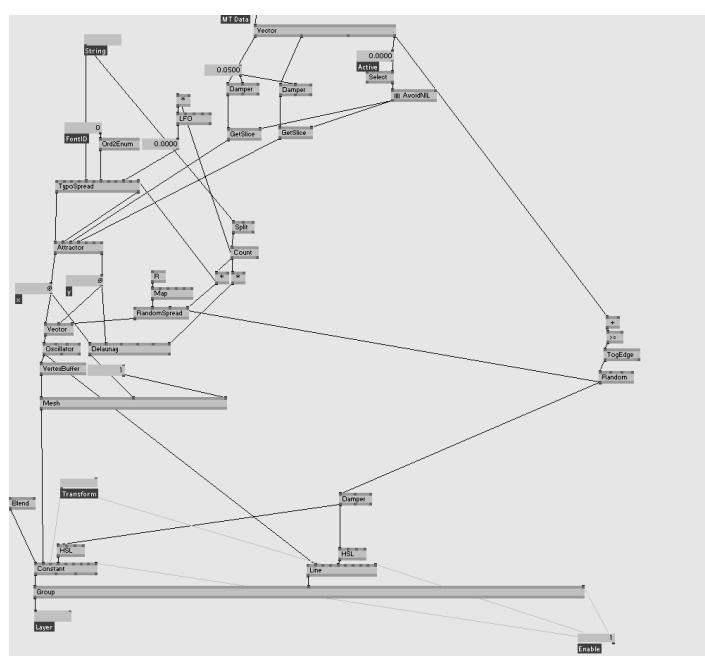
Erfolgt keine Interaktion, ist das Ergebnis aufgrund der hohen Transparenz und des Zielpunktes im Ursprung nicht lesbar. Die BenutzerInnen müssen erst interagieren, um den Text zu sehen.



Delaunay Mesh

In dieser Anwendung wird ein Polygonnetz mit Hilfe des TypoSpreads erzeugt. Die Polygonkoordinaten kommen also nicht aus dem 3d Text, sondern werden eigens zusammengestellt.

Die Grundkoordinaten der Schrift liefert der Typospread. Im Anschluß werden diese mit Hilfe der Delaunay Triangulierung¹ in verschiedene Flächen geteilt. Dies erzeugt die visuelle Erscheinung der Schrift, die sich in diesem Fall nicht wie gewohnt, durch die Füllung ihrer Glyphen beschreibt, sondern durch Flächenansammlungen an detailreichen Stellen ausserhalb des Buchstabens.



Die BenutzerInnen können mit Hilfe des globalen Sliders Zufallswerte auf die z-Koordinaten der Geometrie addieren – ähnlich dem Verfahren in *img 3.2.03* gezeigten Verfahren.

Die Interaktion am Tisch funktioniert erneut mit Attraktoren, die den BenutzerInnen spielerisch die Funktion der Flächenzusammensetzung beibringen, da sich die Flächen bei Veränderung der Attraktoren stets neu bilden.

