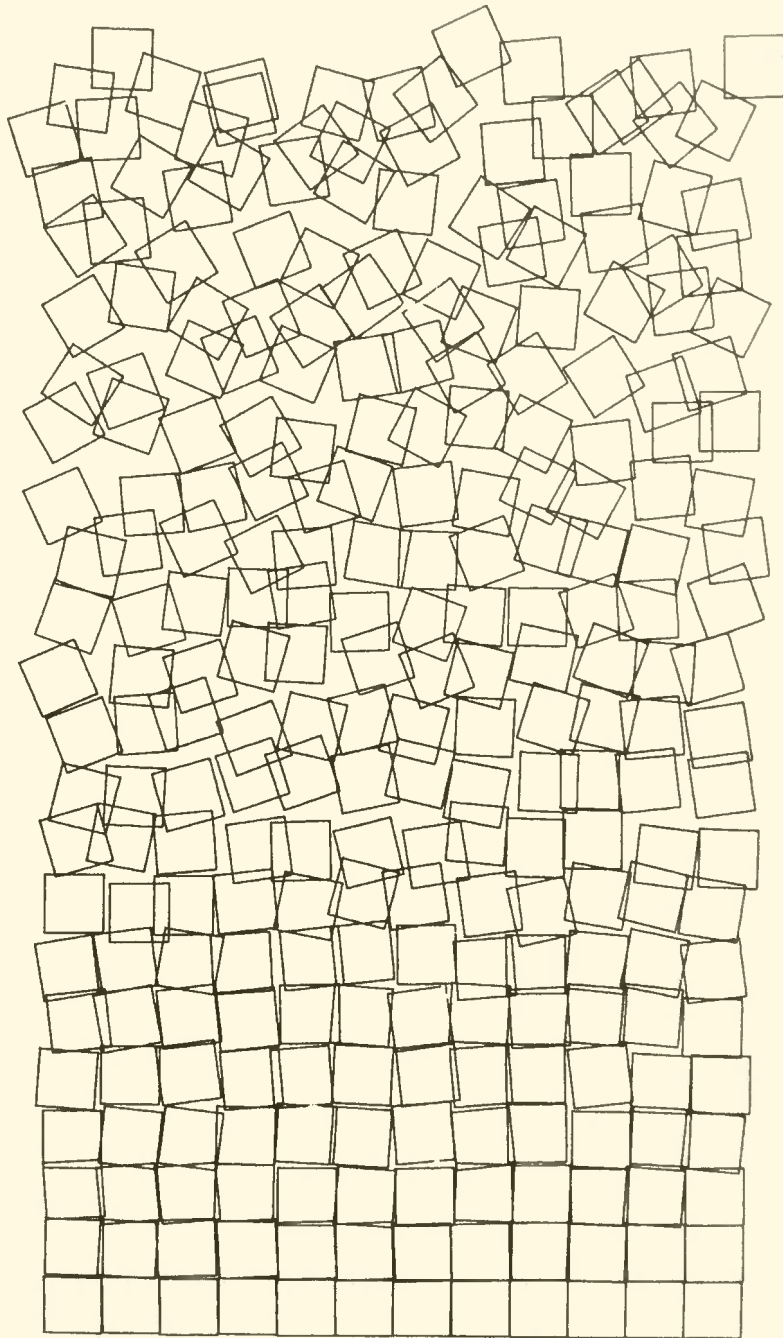


Schotter

60 Jahre Computerkunst



Joachim Wedekind

JOACHIM WEDEKIND

Schotter 60 Jahre Computerkunst

Tübingen 2025

Joachim Wedekind:
Schotter - 60 Jahre Computerkunst

Titelbild: Schotter (Georg Nees, 1965-1969)

2025 Tübingen

© für das Gesamtwerk beim Autor

Grafische Gestaltung und Satz: Joachim Wedekind

Impressum: <http://joachim-wedekind.de/impressum/>

Dieses Werk steht unter einer Creative Commons Lizenz
4.0: Namensnennung, Nicht Kommerziell, Keine Bear-
beitung.

Über diese Lizenz hinausgehende Erlaubnisse können
Sie beim Autor erfragen.



Inhaltsverzeichnis

Vorwort.....	1
Schotter: Das Bild	3
Georg Nees: Der Pionier	5
Schotter: Die Bilder	11
Recoding Schotter	14
Remixing Schotter	26
Schotter: interaktiv & animiert	47
Bilder denken, nicht malen	55
Literatur	56
Bildernachweise	57
Meine Schotter-Programme	58
Zur Person	59

Vorwort

Diese kleine Schrift ist primär einem einzigen Bild gewidmet, der Plotterzeichnung *Schotter* (1965-1969) von Georg Nees, einem der Pioniere der Computerkunst. Dieses Bild ist zu einer Ikone der Algorithmischen und Generativen Kunst geworden. Es wurde vielfach Gegenstand des Recoding, es ist ein gern verwendetes Paradebeispiel in Kursen zur Digitalen Kunst und es war und ist Anregung für das Remixing mit vielen bildnerischen Varianten.

Im Laufe der Jahre habe ich viele Beispiele für das Recoding & Remixing von *Schotter* gefunden, die mir bestätigt haben, welch großen Einfluss die Vorarbeiten von Georg Nees und den anderen Pionieren der frühen Computerkunst auf die Entwicklung der Algorithmischen und Generativen Kunst und damit letztlich auf die aktuellen Formen der Digitalen Kunst gehabt haben und heute noch haben.

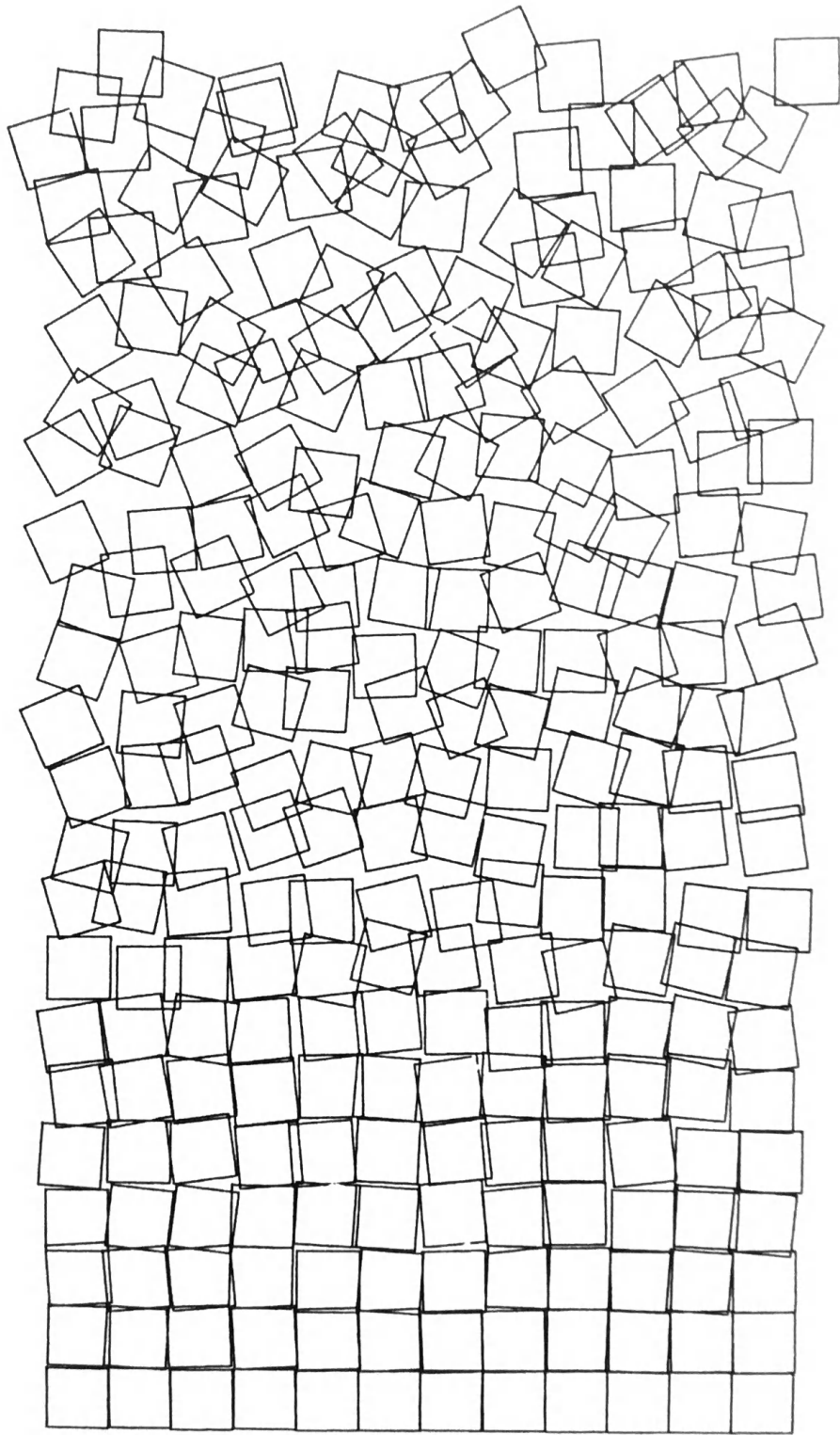
Inzwischen jährt sich zum sechzigsten Mal, dass in den Räumen der Studiengalerie der TH Stuttgart die weltweit erste Ausstellung zur Computerkunst mit Werken von Georg Nees stattgefunden hatte. Ich habe in der etablierten Kunstwelt keine Hinweise geschweige denn Veranstaltungen in Erinnerung an dieses durchaus bedeutsame Ereignis gefunden. Schließlich handelt es sich bei der Computerkunst um den anerkannten Vorläufer der heutigen Digitalen Kunst.

Die vorliegende Schrift ist eine Zusammenfassung meiner wiederholten Beschäftigung mit dem Bild *Schotter*. Sie erscheint 60 Jahre nach der öffentlichen Wahrnehmung der frühen Computerkunst in der Stuttgarter Ausstellung und ist als Hommage an Georg Nees und seinen prägenden Beitrag zur frühen Computerkunst gedacht.

Tübingen, April 2025

Joachim Wedekind

Schotter: Das Bild



*Georg Nees: Schotter (1965-1968, Plotterzeichnung, Tusche auf Papier, 29,7 x 21 cm).
Kunsthalle Bremen - Kupferstichkabinett; aus Groos & Froitzheim, 2016.*

Georg Nees: Der Pionier

Georg Nees (1926 - 2016) war ein deutscher Ingenieur und Mathematiker. Geboren 1926 in Nürnberg, studierte er Mathematik, Physik und Philosophie in Erlangen und Stuttgart.



*Georg Nees, 1986
(Foto: A. Kempkens)*

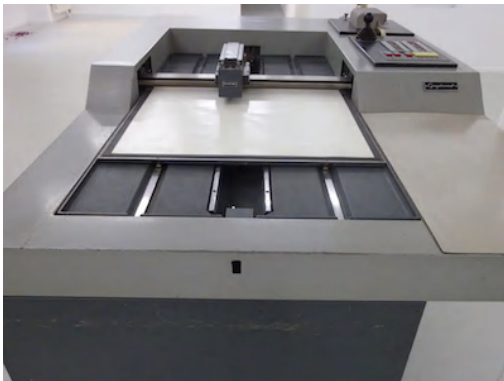
Mit dem Programmieren begann Nees 1959. Ab 1965 arbeitete er bei Siemens in Erlangen und begann dort mit dem Programmieren in der Sprache ALGOL. Dabei ging es etwa bei Architektur-Projekten auch um Computergrafiken.

Während seiner Arbeit bei Siemens begann Nees also sich intensiver mit dem Programmieren von Grafiken zu beschäftigen. Etwa seit 1964 begann er damit, erste eigene Algorithmen in der Sprache ALGOL zu schreiben, mit denen er seine Grafiken mit dem Plotter Zuse Graphomat Z64 zu Papier brachte. Er stellte Muster aus zufällig generierten Zahlen her, die mit dem Plotter auf Papier ausgegeben wurden..

Um diese Art von Grafiken erzeugen zu können, musste er zunächst eigene Grafik-Bibliotheken entwickeln, die die Sprache ALGOL um Zufalls-Funktionen und Funktionen zur Steuerung der Plotter ergänzten. Diese Bibliotheken waren damals notwendige Voraussetzung für seine bildnerischen Arbeiten. Dasselbe galt für die benötigten Zufallszahlengeneratoren.

Die Entwicklung eines Bilds wie *Schotter* war für die damalige Zeit eine echte Pionierleistung. Da es noch keine Monitore gab, auf denen eine Visualisierung seiner Projekte hätte dargestellt werden können, musste Nees sich auf seine Vorstellungskraft verlassen, welches Ergebnis mit seinen Algorithmen zu erreichen war.

Nees programmierte seine frühen Grafiken auf Siemens-Systemen, wie dem System 4004, ein Großrechner, den Siemens ab 1968 selbst produzierte. Für den Ausdruck der Grafiken standen Nees Plotter zur Verfügung, die damals vor allem in technischen und naturwissenschaftlichen Bereichen eingesetzt wurden; konkret war es der *Graphomat Z 64* der Firma Zuse:



ZUSE Graphomat Z 64, Zeichentisch (links); Steuerelektronikschrank (unten). Fotos: IW, 2015



Auf der Hannovermesse 1961 stellte der Computerpionier Konrad Zuse mit dem Zeichentisch Graphomat Z64 einen Flachbett-Plotter vor, der mit Lochkarten gesteuert wurde. Er zeichnete sich durch eine hohe Zeichengenauigkeit aus und konnte mit bis zu vier Farben Punkte, Kurven und Symbole zeichnen. Dank hochgenau arbeitender Planetengetriebe konnte das Gerät mit 16 verschiedenen Geschwindigkeiten angesteuert werden. Die Seriengeräte gab es für die Ausgabeformate 550 x 600mm und 1200 x 1400mm. Sie wogen ca. 1400 kg; die Bilder oben können nur einen groben Eindruck von der Größe und Masse des Graphomat Z64 geben.

Im Grunde wurde mit den Plottern eine geradezu konventionelle Bilderzeugung betrieben: Die Plotter bewegen einen Stift über die Oberfläche eines Papierbogens. Das Ergebnis sind Vektorgrafiken, keine Rastergrafiken, wie wir sie von unseren modernen Bildschirmen kennen. Plotter arbeiten nicht zeilenweise wie Drucker, sondern zeichnen die Linien von Vektorgrafiken nach. Vektorgrafiken sind ohne Qualitätsverlust beliebig skalierbar und vermeiden dadurch die technisch bedingte Limitierung von Rastergrafiken hinsichtlich der Bildauflösung und Bildgröße. Sie eignen sich deshalb auch besonders für große und komplexe Grafiken.

Bei den Stiftplottern wird eine Stifthalterung auf zwei Achsen bewegt. Zusätzlich kann der Stift gehoben und gesenkt werden. Dabei werden ein einzelner Farbstift oder mehrere an einer Halterung befestigte Stifte mit verschiedenen Farben durch Motoren über das Papier gefahren und zeichnen dabei die Linien. Flächen sind so nur durch enge Schraffuren zu erzielen. Für jede Farbe und für jede Liniestärke muss ein anderer Stift verwendet werden. Aufgrund der mechanischen Bewegung der Stifte ist die Erstellung sehr langsam. Lange Zeit war dies dennoch der einzige Weg vektorbasierte Bilder auch in großen Formaten zu erzeugen.

Für das Zeichnen eignen sich Filz- und Tuschestifte sowie Kugelschreiber, die auch unterschiedliche Strichbreiten erlauben. Anfangs waren meist nur einfarbige Zeichnungen möglich, da ein Farbwechsel mit dem Unterbrechen eines laufenden Programms und dem erst dann möglichen Wechsel des Stiftes verbunden war. Später war auch ein automatischer Farbwechsel über Farbkarussells möglich.

Heute sind Plotter in ihren ursprünglichen Einsatzgebieten weitgehend überflüssig geworden. Ihre Aufgaben werden von Laser- oder Tintenstrahldruckern entsprechender Größe übernommen. Dennoch erleben Plotter derzeit eine Art Renaissance, weil sie heute oft erweiterte Anwendungsbereiche und Gerätekombinationen bieten. Es gibt erschwingliche Geräte, mit denen sogar Bildgrößen bis zum Format A1 möglich werden.

Bei aller Präzision haftet den Plottern etwas nicht-maschinenhaftes an. Plotterbilder sind in der Regel keine 1:1-Abbilder der am Bildschirm gezeigten Bildergebnisse eines Programms. Die Bewegung des Schlittens mit den Zeichenstiften, seine Beschleunigung, das Abbremsen, das Anheben und Absenken der Stifte sowie die Eigenschaften von Papier und Tinte sorgen oft für das Auftreten von Artefakten. Im Grunde ist deshalb jedes Bild ein Unikat, das eine Nähe zur herkömmlichen manuellen Arbeit mit Stift und Papier aufweist.

Soviel zu den technischen Aspekten der Arbeiten von Georg Nees, die einen wichtigen Hintergrund für seine Rolle in der *Computerkunst* bilden. Im Jahr 1969 promovierte Nees in Philosophie über *Generative Computergraphik* (Nees, 1969). Sein Doktorvater war Max Bense (1910-1990), Professor für Philosophie der Technik, Wissenschaftstheorie und mathematische Logik an der Technischen Hochschule Stuttgart. Bense prägte den Begriff der *Informationsästhetik*. In seinem Umfeld hatte sich seit Anfang der 1960er Jahre eine Gruppe von Promovenden formiert, die die Informationsästhetik vertieften. Zu ihnen zählten u.a. Rul Gunzenhäuser, Frieder Nake und Helmar Frank.

Nees versucht in seiner Dissertation die grundlegenden Begriffe und philosophischen Aspekte der Ästhetik von Bense zu konkretisieren. Deshalb behandelt er das Thema Computergraphik sowohl aus einer computerwissenschaftlichen als auch aus einer philosophisch-ästhetischen Perspektive. Er führt mit konkreten Beispielen in diese Denkweise ein und veranschaulicht das Zusammenspiel von prozeduraler Abstraktion und (kontrolliertem) Zufall anhand der Entwicklung exemplarischer Programme. Sein Anwendungsfeld ist die *Computerkunst*¹, ein typisches Ergebnis ist das Bild *Schotter*.

Bereits als Student von Max Bense wurde Nees eine von Bense organisierte erste Ausstellung seiner Computerkunst in Stuttgart ermöglicht. Im Februar 1965 fand damit an der Universität Stuttgart die weltweit erste Ausstellung zur Computerkunst statt - mit Bildern, die Nees selbst *Ästhetische Objekte* nannte. Es gab eine Vernissage, bei der Bense und Nees sprachen und es gab Pressereaktionen. Auch einige etablierte Künstler waren anwesend, wie Heinz Trokes, Anton Stankowski und Herbert W. Kapitzki. Einer der anwesenden Künstler fragte: "*Aber können Sie ihren Computer auch dazu bringen, so zu malen wie ich?*" Und Georg Nees soll geantwortet haben: „*Selbstverständlich. Falls Sie mir sagen, wie Sie es machen.*“

„*Genial*“ findet der Mathematiker Nake, der zeitgleich mit Georg Nees in Stuttgart an der Produktion künstlerischer Computergraphiken arbeitete², diesen Dialog, denn er sei so typisch für die Anfänge der digitalen Ästhetik. Laut Nake ist es falsch dem Computer künstlerisches Vermögen zuzuschreiben. Aber erstmalig sei es möglich geworden, mittels Zufallsgeneratoren Ideen auf eine Art umzusetzen,

¹ H.W. Franke definiert Computerkunst wie folgt (Franke, 1971, S. 7 f.): „*Unter Computerkunst soll jedes ästhetische Gebilde verstanden werden, das auf Grund von logischen oder numerischen Umsetzungen gegebener Daten mit Hilfe elektronischer Automaten entstand.*“ Sie entsteht demnach immer mit Hilfe digital gesteuerter Geräte, d.h. mit Programmen zur Erzeugung der darzustellenden Daten, den Computern, auf denen die Programme laufen und die Ausgabegeräte, mit denen die Daten als wahrnehmbare Artefakte dargestellt werden.

² Beide stellten im November 1965 zusammen in der Galerie Niedlich (Stuttgart) aus.

die für den Urheber des Programms unvorhersehbar war. Für Nake verändert dies den Bildbegriff in der Kunst - auch wenn in Kunstkreisen die Werke als Krakeleien abgelehnt wurden.

In der frühen Computerkunst nimmt Georg Nees einen ganz besonderen Platz ein. Er war nicht nur einer der ersten, der solche Werke schuf und ausstellte, er war sich auch von Anfang an des innovativen Charakters seiner Grafiken bewusst: *„Als ich sah, wie eine Figur nach der anderen unter dem Stift hervorspringt, lief es mir kalt den Rücken hinunter. Ich dachte: Hier ist etwas, das nicht wieder verschwinden wird.“* (Nees, 2005).

Georg Nees verstand das Arbeiten mit Computer und Plotter als *ästhetisches Laboratorium* (Nees, 1995, S. 7). Mit dem Computer steht ihm eine Versuchswerkstatt zur Verfügung, die es ihm erlaubt, grafisch zu experimentieren *„bis es uns gelingt, gezielt - oft genug aber auch als Ergebnis eines glücklichen Zufalls - ästhetische Objekte zu erzeugen“*. Nees (a.a.O., S. 183) hat das Vorgehen einmal als *Gestaltfallenstellen* charakterisiert: *„Irgendwann wird man durch einen außergewöhnlichen Fang belohnt“*.

Nees programmierte seine Werke bewusst unter ästhetischen Gesichtspunkten und verstand sie als Modelle des künstlerischen Produktionsprozesses. Er war deshalb ein Pionier der Computerkunst, der nicht nur prägende Grafiken gefunden hat, sondern systematisch die Redundanzen in zufallsgesteuerten Bildern untersuchte. Georg Nees entwarf 1968 zudem die erste durch einen Computer generierte Skulptur.

Bekannte Konzeptkünstler wie Sol LeWitt befassten sich mit regelbasierter Kunst, Op-Art Künstler beschäftigten sich mit wahrnehmungspsychologisch mehrdeutigen Formen und arbeiteten dazu auch mathematisch motivierte Kunstwerke aus. Die frühen Computerkünstler wollten ihrerseits diese beiden Ansätze vereinen und sie mittels neuer Technologie zum Ausdruck bringen. In diesem Sinne versuchte Georg Nees mit seinen Werken zu zeigen, dass sich Wissenschaft, mathematische Berechnungen, Algorithmen und Technik zu einem Kunstwerk vereinen lassen.

Besonders die Konzeptkunst und Computerkunst werden gern als eng miteinander verwandt beschrieben. Nake charakterisiert dieses Verhältnis allerdings etwas differenzierter wie folgt (Nake, 2021, S. 64):

„In ‚The idea becomes a machine that makes the art‘, hatte Sol LeWitt 1967 geschrieben: ‚Die Idee wird zu einer Maschine, die die Kunst macht.‘ Arg verspätet, aber immerhin, möchte ich antworten: ‚Exactly this, dear Sol LeWitt, has happened some years before you wrote it.‘

Denn seit etwa 1962 hatten in den USA und in West-Deutschland ein paar Computerexperten damit experimentiert, ihre Ideen zu Computerprogrammen werden zu lassen, die Zeichnungen an die Wände von Galerien bringen sollten, also nichts anderes als Kunst zu machen. [...] Die Maschine, die den Konzeptionierern ein Gedanke war, war den Computationierern handfeste Wirklichkeit.“

Wer die Überlegungen von Georg Nees im Einzelnen nachvollziehen möchte, sollte unbedingt seine Dissertation lesen und studieren. Sie ist auch als Buch *Generative Computergraphik* (1969) erschienen³. Die bewusste Gegenüberstellung zentraler Aspekte der Informationsästhetik von Bense mit der Programmierung seiner Projekte und deren bildnerischen Ergebnisse ist bis heute eine lohnende und anregende Lektüre.

Die Erfahrungen, die Nees bei der Programmierung seiner ästhetischen Objekte gewonnen hat, sind außerdem in einer systematischen Einführung in die Welt der Computerkunst zusammen gefasst. In *Formel, Farbe, Form. Computerästhetik für Medien und Design* (Nees, 1995) eröffnet er mit aktuellerer Software - der Programmiersprache BASIC - und sehr vielfältigen Beispielen einen praktischen Zugang zur selbständigen Erzeugung interessanter und ansprechender Grafiken.

Sein Buch zeigt „... mit wie einfachen Programmierkenntnissen wirklich alle, ob professionell oder aus Liebhaberei, reiche Schätze von Bildern, Farben, Designs und Strukturen heben können [...]. Die ersten Schritte sind so leicht: Man geht von Abbildungen aus, die einem besonders auffallen. Dann werden Neugier, genaueres Studium des Textes und vielleicht auch ein bereitstehender Computer den Weg in die verlockende Formenwelt stufenweise bahnen.“ (Nees, 1995, S. VII)

Tatsächlich wird damit auch mein persönlicher Weg in die Computerkunst von Nees sehr zutreffend beschrieben. Ohne seine Anregungen würde es meine eigenen Beiträge zur Computerkunst (Ausstellungen, Vorträge und Schriften) jedenfalls in der vorliegenden Form nicht geben.

³ Ein Nachdruck mit einleitenden Texten von Herrmann und Nees ist 2006 erschienen und auch als E-Book erhältlich.

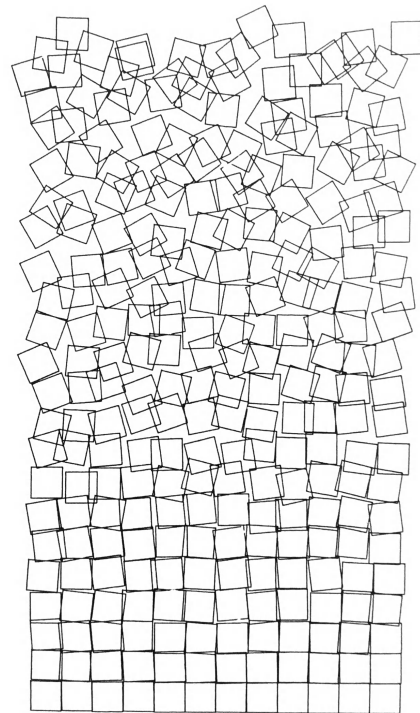
Schotter: Die Bilder

Mir ist das Bild *Schotter* zum ersten Mal begegnet, als ich Programmierübungen für eine Einführung in das Programmieren mit der Schildkrötengrafik gesucht habe⁴. Entdeckt hatte ich es - dort *Schotterhaufen* genannt - in einer Einführung in die Programmiersprache Logo, anfangs ohne zu wissen, dass es ein Werk von Nees war⁵. Bei mir entstand daraus jedenfalls meine erste eigenständige Umsetzung dieses Bildes.

Erst Jahrzehnte später, als ich begonnen hatte, mich intensiv und aktiv mit der Computerkunst auseinander zu setzen, fand ich das Original des Bildes und die Informationen zum Urheber, dem Ingenieur und Mathematiker Georg Nees (Wedekind, 2018, S. 149)

Ausgangspunkt der Darstellung in diesem Kontext ist das Originalbild von *Schotter*, wie es in der Dissertation von Georg Nees (Nees, 1969, S. 242) zu finden ist. Diese Version gehört auch zum Bestand des Kupferstichkabinetts der Kunsthalle Bremen und wurde 2016 in der Ausstellung *[un]erwartet* im Kunstmuseum Stuttgart gezeigt (Groos & Froitzheim, 2016).

Es gibt aber wohl weitere Varianten von dem Bild *Schotter* mit jeweils leicht unterschiedlichen Abmessungen. So besitzt etwa das Victoria and Albert Museum (Kensington, GB)⁶ eine Version, datiert mit 1968-1970. Eine andere, datiert mit 1968 besitzt The Anne and Michael Spalter Digital Art Collection⁷.



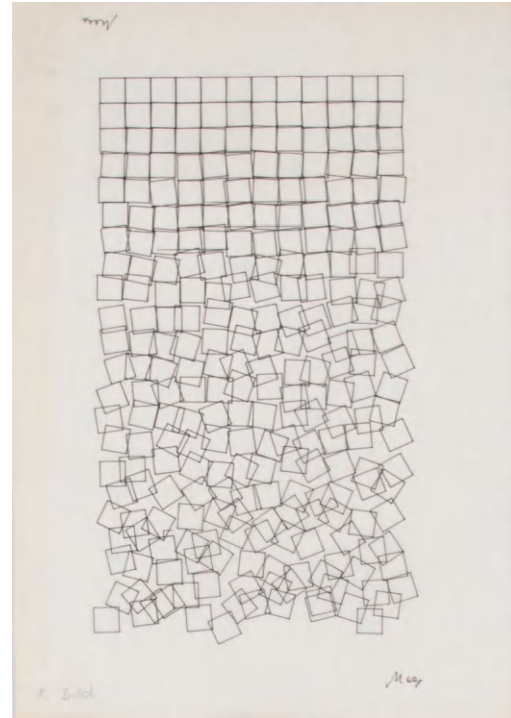
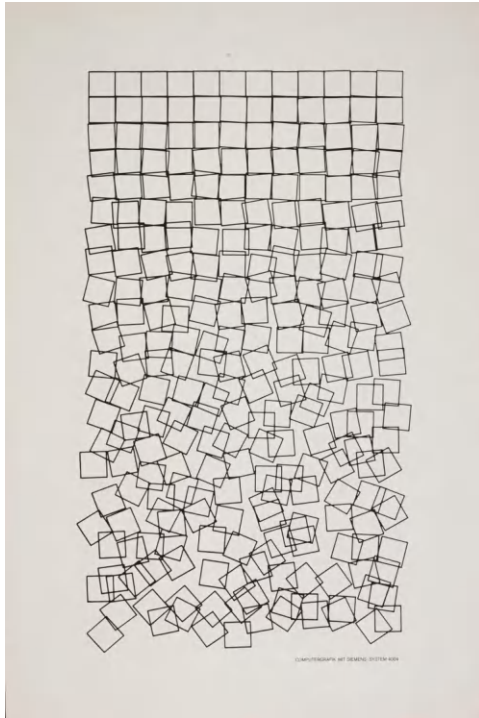
Georg Nees: *Schotter* (aus Nees, 1969, Bild 38, S. 242)

⁴ Bei der *Schildkrötengrafik* handelt es sich um die Grafikkomponente der Programmiersprache Logo. Eine entsprechende Version haben wir im Rahmen eines Fernstudienprojekts *Lehren und Lernen mit dem Computer* entwickelt und vertrieben (Rauch & Wedekind, 1989).

⁵ Das Buch von Nees taucht zwar ohne Nennung von Verlag und Jahrgang in den Literaturangaben dieses Buches auf (Stein, 1984), aber im Text wird die Herkunft der sich über immerhin 70 Seiten erstreckenden Nachprogrammierung mehrerer Nees-Beispiele mit keinem Wort erwähnt.

⁶ <https://collections.vam.ac.uk/item/O221321/schotter-print-nees-georg/>

⁷ <https://spalterdigital.com/artworks/shotter-gravel-stones/>



Georg Nees: Schotter: V&A Museum (links); Spalter Collection (rechts)

Das Schwarz-Weiß-Bild setzt sich aus lauter Quadraten mit einer festen Kantenlänge zusammen. Die Bildstruktur ist im Grunde sehr einfach: *Schotter* besteht aus 22 Reihen mit jeweils 12 Quadraten. In der ersten Reihe liegen die Quadrate unverdreht dicht nebeneinander. Von Reihe zu Reihe werden die Position und die Ausrichtung (Rotation) der Quadrate leicht verändert. Dabei steigert sich mit jeder Reihe allmählich das Ausmaß der Zufälligkeit bei der Rotation und der horizontalen sowie vertikalen Platzierung der Quadrate. Die Ordnung der Quadrate wird dadurch zunehmend gestört. Alle Quadrate sind transparent, weshalb die entstehenden Überlappungen immer sichtbar bleiben.

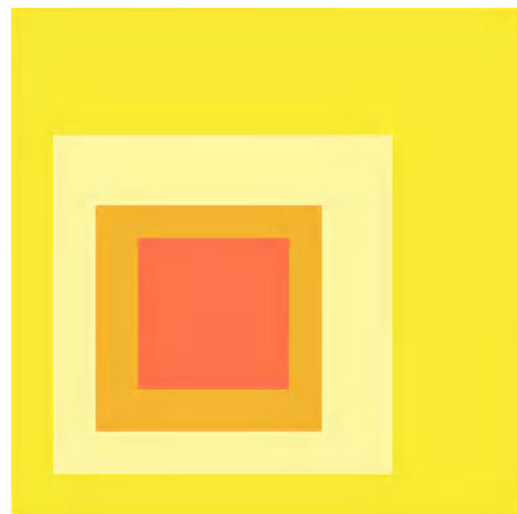
In der Dissertation von Nees nimmt die Unordnung der Quadrate von unten nach oben zu, bei den oben genannten Sammlungsstücken ist es genau umgekehrt. Ein Zeitgenosse von Nees erzählte die Anekdote, nach der Georg Nees auf die Frage nach der richtigen Ausrichtung des Kunstwerks antwortete, dass es ihm egal sei, ob das Werk mit kaskadierenden oder aufsteigenden Quadraten dargestellt werde. Ich selber bevorzuge die kaskadierende Version, weil meine Vorstellung dabei ist, wie Schottersteine von einem Laster gekippt werden.

Das Bild *Schotter* ist ein Paradebeispiel für das Zusammenspiel von Ordnung und Unordnung durch (gelenkten) Zufall. Es visualisiert direkt deren Wechselwirkung. Ein weiteres zentrales Element ist die Wiederholung, die sich in der wiederkehrenden Abfolge gleichartiger Bildelemente - wie hier von den Quadraten - manifestiert. Solche repetitiven Techniken sind kein Spezifikum der Arbeit am Computer.

Frieder Nake diskutiert die kulturelle Bedeutung solcher basalen Möglichkeiten der Computernutzung als Werkzeug und Medium (Nake, 1995). Er setzt die Computerkunst an konkreten Beispielen in Bezug zur Pop Art, der konkreten Kunst und dem Konstruktivismus. Als ein typisches Beispiel dafür dient ihm die Serie *Homage to the Square* von Josef Albers, die dieser ab 1949 in zahllosen Varianten erstellt hat und von denen jede ein Original darstellt. Für Nake ist es das vielleicht einfachste Beispiel der Kunstgeschichte, das in seiner Systematik mit einem kombinatorischen Programm vollständig ersetzt werden kann. Sein ironisches Fazit ist (was meines Erachtens wohl für viele Werke moderner Kunstrichtungen gelten dürfte), dass

„[...] die Kunst an den Albersschen Quadraten nicht im Schema liegt. Sie muss in seiner speziellen Wahl der Farben liegen. Ansonsten aber ist diese pädagogische Übung ‚Kunst‘ nur dadurch, dass die Kunstgeschichte diese Bilder in die Museen hängt und in die Kataloge aufnimmt. [...] Der soziale Prozess der Rezeption macht ein Werk zum Kunstwerk, nicht der Schaffensprozess des Künstlers.“

Nakes Vorschlag eines kombinatorischen Programms zur Erzeugung der verschachtelten Quadrate („ein Schema für den Anfangsunterricht in Informatik“, Nake, a.a.O.) ist tatsächlich äußerst einfach umzusetzen⁸.



Remixing von Josef Albers: Homage to the Square, JW 2015

⁸ Mein Remixing der Quadrat-Serie von Josef Albers finden Sie hier: [Hommage to the Square: \[Programm 1\]](#)

Recoding Schotter

Georg Nees hat als einer der ganz Wenigen seinen Algorithmus für das Bild *Schotter* dokumentiert (Nees, 1969, S. 241 u. 186). Das dient als Vorlage für das *Recoding & Remixing* des Bildes. Unterschiedliche Vorgehensweisen sind dabei möglich, worauf ich hier eingehen möchte. Beginnen wir also mit seinem Originalprogramm:

```

1 'BEGIN' 'COMMENT' SCHOTTER.,
2 'REAL' R, PIHALB, PI4T.,
3 'INTEGER' I.,
4 'PROCEDURE' QUAD.,
5 'BEGIN'
6 'REAL' P1, Q1, PSI, 'INTEGER' S.,
7 JE1.=5*I/264., JA1.= -JE1.,
8 JE2.=PI4T*(1+I/264.),
9 JA2.=PI4T*(1-I/264.),
9 P1.=P+5+J1., Q1.=Q+5+J1.,
  PSI.=J2.,
10 LEER(P1+R*COS(PSI),
  Q1+R*SIN(PSI)),
11 'FOR' S.=1 'STEP' 1 'UNTIL' 4 'DO'
12 'BEGIN' PSI.=PSI+PIHALB.,
13 LINE(P1+R*COS(PSI),Q1+R*SIN(PSI))
14 'END', I.=I+1
15 'END' QUAD.,
16 R.=5*1.4142.,
17 PIHALB.=3.14159*.5., PI4T.=PIHALB*.5.
18 I.=0.
19 SERIE(10.0,10.0,22,12,QUAD)
20 'END' SCHOTTER.,

1 'REAL' P, Q, P1, Q1
2 ,XM, YM, HOR, VER
3 ,JL1, JRE, JUN, JOB
4 „
5 'INTEGER' I, M, N, T
6 „
7 'PROCEDURE' SERIE
  (QUER,HOCH,XMAL,YMAL,FIGUR),
8 'VALUE' QUER, HOCH, XMAL, YMAL.,
9 'REAL' QUER, HOCH.,
10 'INTEGER' XMAL, YMAL.,
11 'PROCEDURE' FIGUR.,
12 'BEGIN'
13 'REAL' YANF.,
14 'INTEGER' COUNTX, COUNTY.,
15 P.= -QUER*XMAL*.5.,
16 Q.=YANF = -HOCH*YMAL*.5.,
17 'FOR' COUNTX.=1 'STEP' 1 'UNTIL' XMAL
  'DO'
18 'BEGIN' Q.=YANF.,
19 'FOR' COUNTY.=1 'STEP' 1 'UNTIL' YMAL 'DO'
20 'BEGIN' FIGUR., Q.=Q+HOCH
21 'END', P.=P+QUER
22 'END',
23 LEER(-148.0,-105.0), CLOSE.,
24 SONK(11),
25 OPEN(X,Y)
26 'END' SERIE.,
  
```

Von Nees wird der Code wie folgt kommentiert (Nees, 1965, S. 241):

„[Schotter] wird durch einen Aufruf der Prozedur SERIE erzeugt [...]. Zur Genese der Elementarfigur, die in dem von SERIE gesteuerten Kompositionsprozeß vervielfacht wird, dient die parameterlose Prozedur QUAD. In den Zeilen 4 bis 15 des Generators wird QUAD vereinbart, diese Prozedur zeichnet Quadrate konstanter Seitenlänge, jedoch zufälliger Lage und Winkelstellung. Man erkennt aus den Doppelzeilen 9 und 10, daß die Position des Einzelquadrats vom Zufallsgenerator J1, die Winkellage von J2 beeinflusst wird. Die sukzessive verbreiterte Streuung der relativen Ortskoordinaten P und Q und des Lagewinkels PSI des einzelnen Quadrats wird durch einen Zähler I gesteuert, der bei jedem Aufruf von QUAD weitergeschaltet wird (siehe Zeile 14).“

Ohne an dieser Stelle auf Einzelheiten einzugehen, ist im Programm SCHOTTER (links, blau markiert) zu sehen, dass dort eine Prozedur QUAD (rot mar-

kiert) definiert wird, die beim Aufruf der Prozedur SERIE (rechts, grün markiert) übergeben wird. In der Prozedur SERIE ist wiederum ersichtlich, dass in zwei verschachtelten Schleifen (grau markiert) die von der Prozedur QUAD erzeugten Quadrate Reihe für Reihe mit zufälliger Lage und Winkelstellung dargestellt werden.

Durch die Prozedur QUAD werden die elementaren Elemente des Bildes, die Quadrate erzeugt. Jedes Quadrat besitzt eine feste Kantenlänge, eine bestimmte Position und Ausrichtung. Die genauen Werte für die Position und die Ausrichtung des jeweils als nächstes zu zeichnenden Quadrats wird über Zähler ermittelt, die bei jedem Durchlauf der Schleifen um den Wert Eins erhöht werden. Mit der Zunahme der Zähler nehmen dann auch die Abweichungen zu.

Mit seinem Code für das Bild *Schotter* hat Nees die Vorlage für das *Recoding* seines Bildes zur Verfügung gestellt. Den Versuch, Werke der frühen Computerkunst zu Recoden, gab es schon vor dem Ende der ersten Blütezeit der Computerkunst Mitte der 80er Jahre. Ein Grund dafür war und ist das Interesse, aus den vorliegenden Arbeiten zu lernen und Anregungen zu bekommen - ein Vorgehen, dass in der Ausbildung zur klassischen Malerei mit der Kopie bekannter Werke zu Übungszwecken eine lange Tradition hat. Bei der Computerkunst betrifft die Möglichkeit des Lernens sowohl die technischen Aspekte als auch die Bildkomposition.

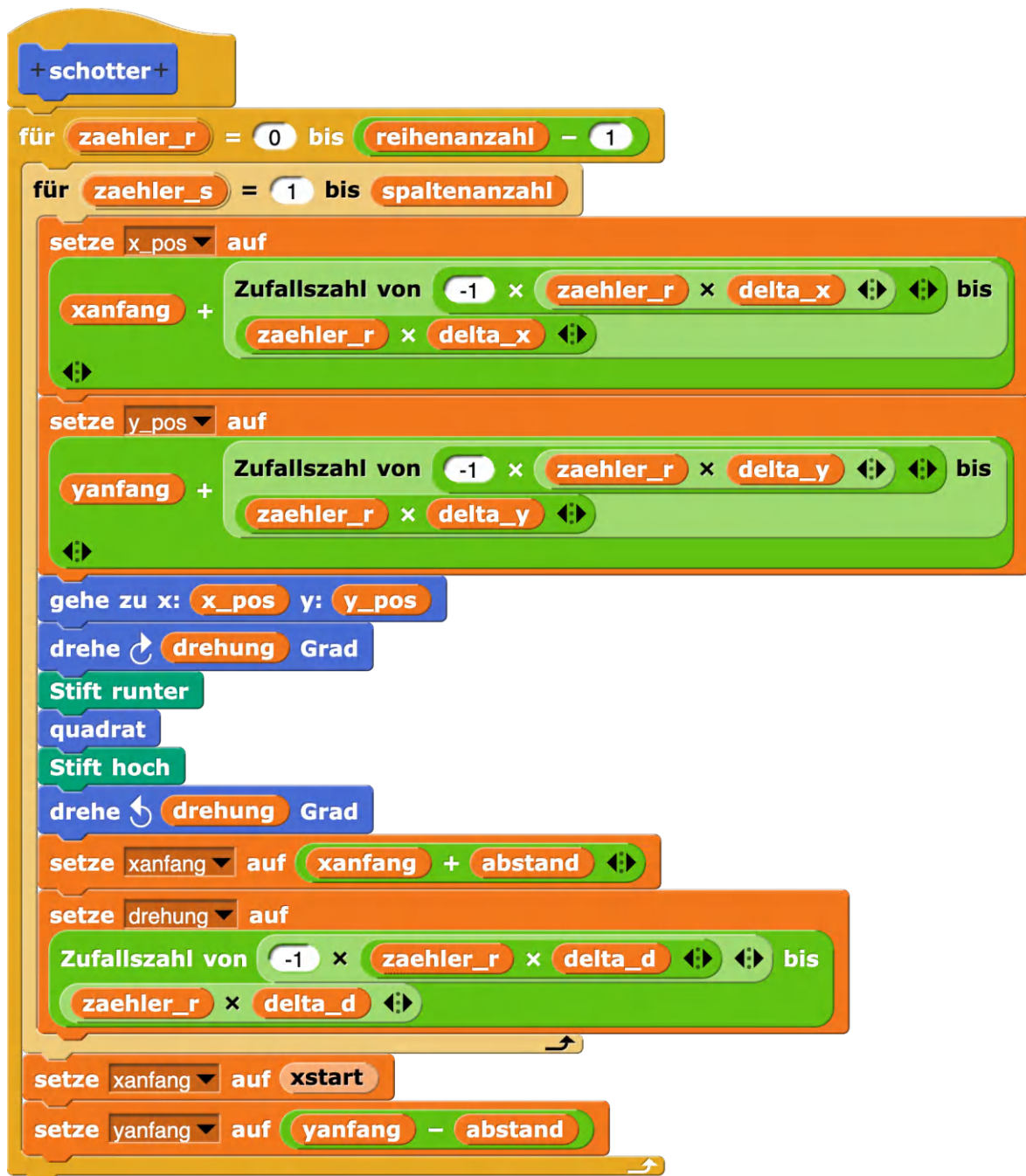
Mein erster eigener Versuch zum Recoding von *Schotter* war 1988 mit der Schildkrötengrafik, was ich 2014 mit ACSLogo⁹ wieder aufgegriffen und wiederholt habe. Seit ich mit meinen eigenständigen Arbeiten zur Computerkunst begonnen habe, verwende ich die visuelle Programmierumgebung Snap!¹⁰, die sich für mich als ideales Werkzeug herausgestellt hat (ausführlich dazu in meinem Lehrbuch *Codierte Kunst*, Wedekind, 2018).

Wenn wir uns eng an den Code von Nees halten und dies in Snap! umsetzen, erhalten wir das nebenstehende Programm. Unterschiede finden sich in der Bezeichnung der Kenndaten:

- **xstart, ystart**: Anfangspunkt des Zeichenvorgangs
- **xanfang, yanfang**: Anfangspunkt einer Quadratreihe

⁹ Diese Logo-Version - von Alan Smith 2002 vorgestellt - ist bis heute als ACSLogo Version 1.6.0.2 (nur für MacOS X) zum Download verfügbar (<https://www.alancsmith.co.uk/logo/>).

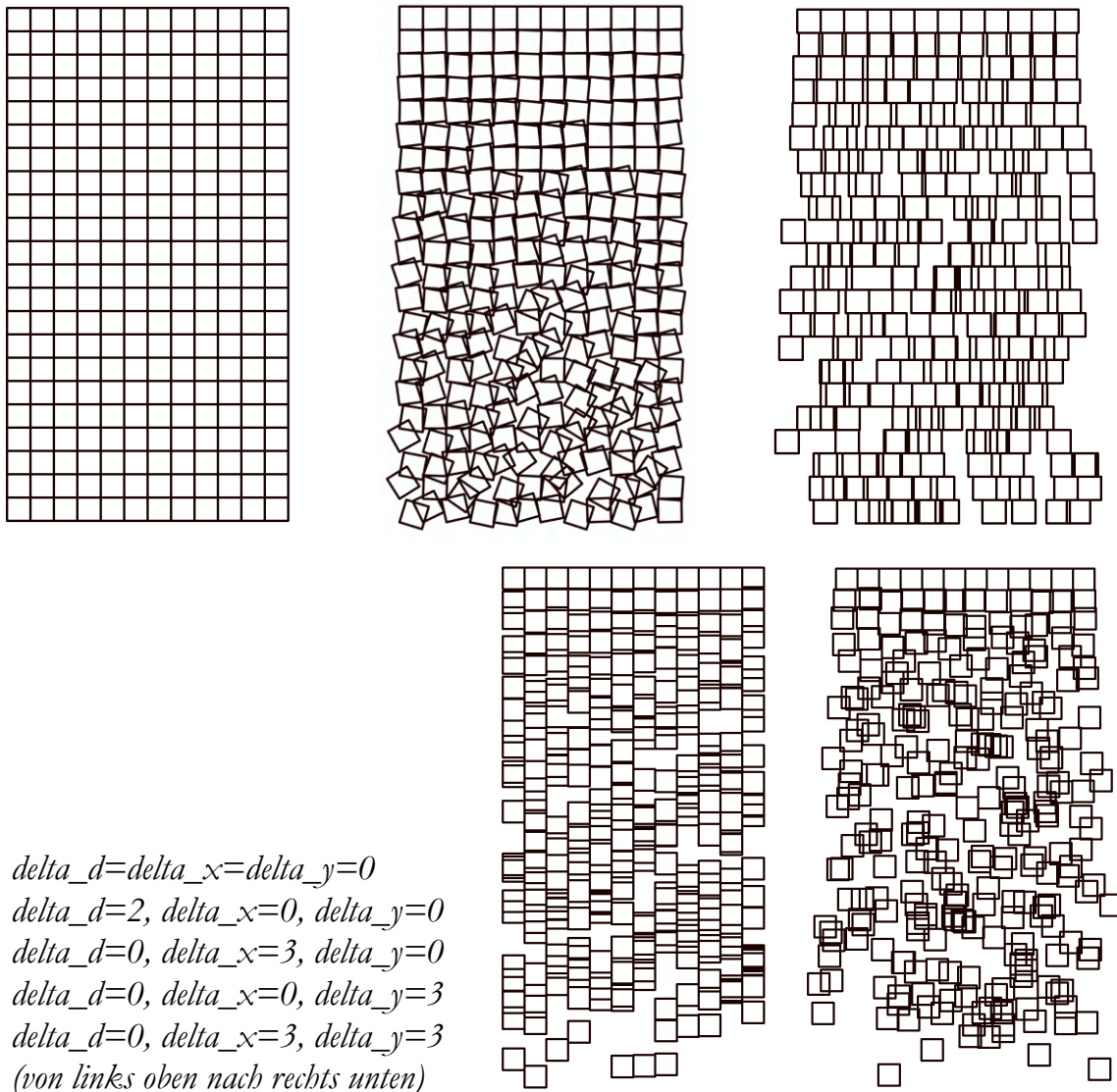
¹⁰ Snap! (<https://snap.berkeley.edu/>) wird von Brian Harvey und Jens Mönig entwickelt und im Internet frei zur Verfügung gestellt. Das System läuft im Browser, benötigt deshalb keine Installation und läuft auf allen gängigen Plattformen.



- **xpos, ypos**: aktuelle Position eines einzelnen Quadrats
- **spaltenanzahl, reihenanzahl**
- **laenge**: Seitenlänge der Quadrate
- **abstand**: horizontaler und vertikaler Abstand zwischen den Quadraten
- **drehung**: Drehwinkel eines Quadrats
- **delta_d**: Grad der Veränderung des Drehwinkels von Reihe zu Reihe
- **delta_x, delta_y**: Grad der Veränderung der x- und y-Position der Quadrate von Reihe zu Reihe

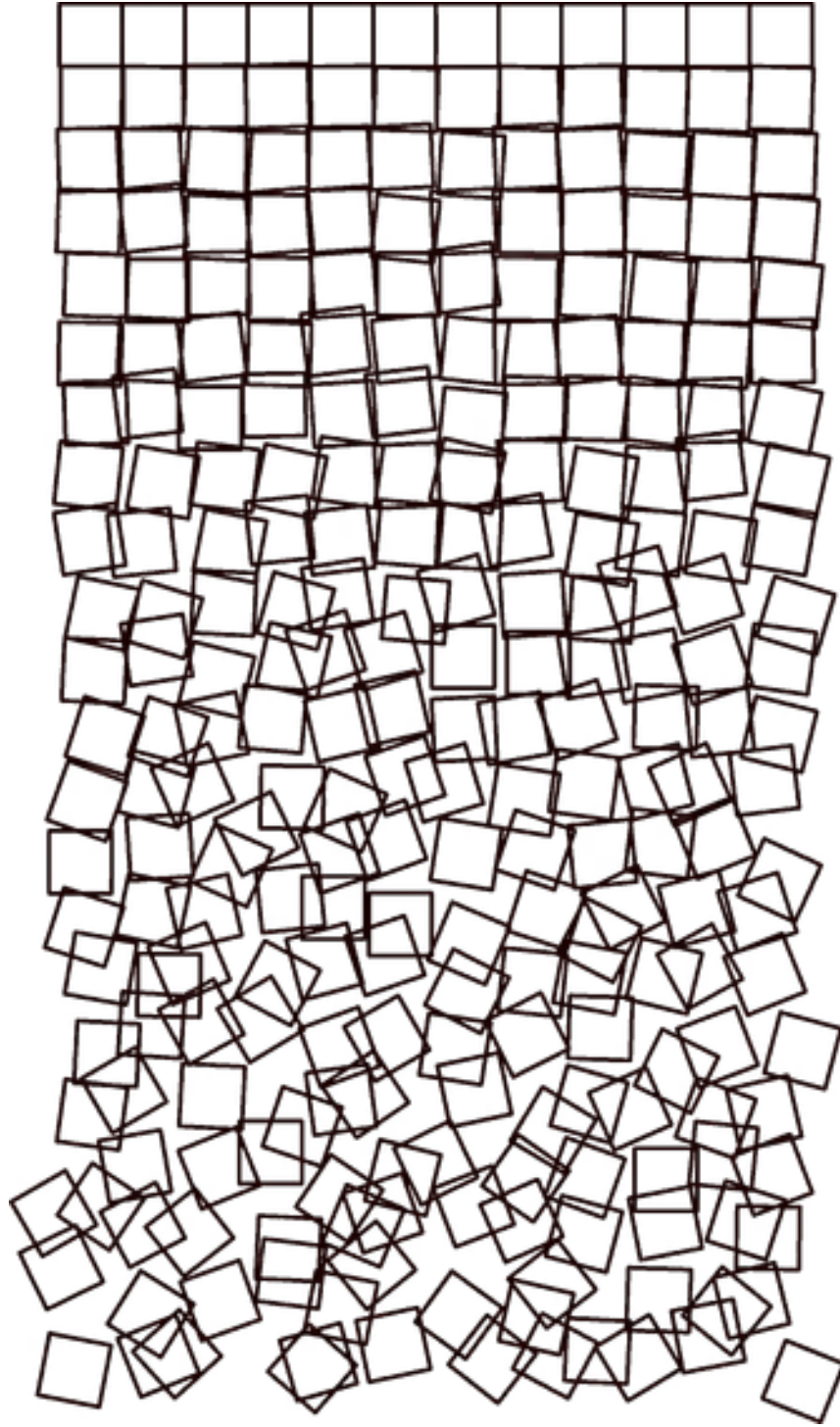
Nach Programmstart werden die Anfangswerte dieser Kenndaten festgelegt und dann der Block **schotter** ausgeführt. Trotz der Abweichungen im Detail kann sich daraus ein dem Original sehr ähnliches Bild ergeben.

Die einfache Änderung der Kenndaten **delta_d**, **delta_x** und **delta_y** erlaubt das gezielte Testen der Zufallseinflüsse auf die Bildstruktur.

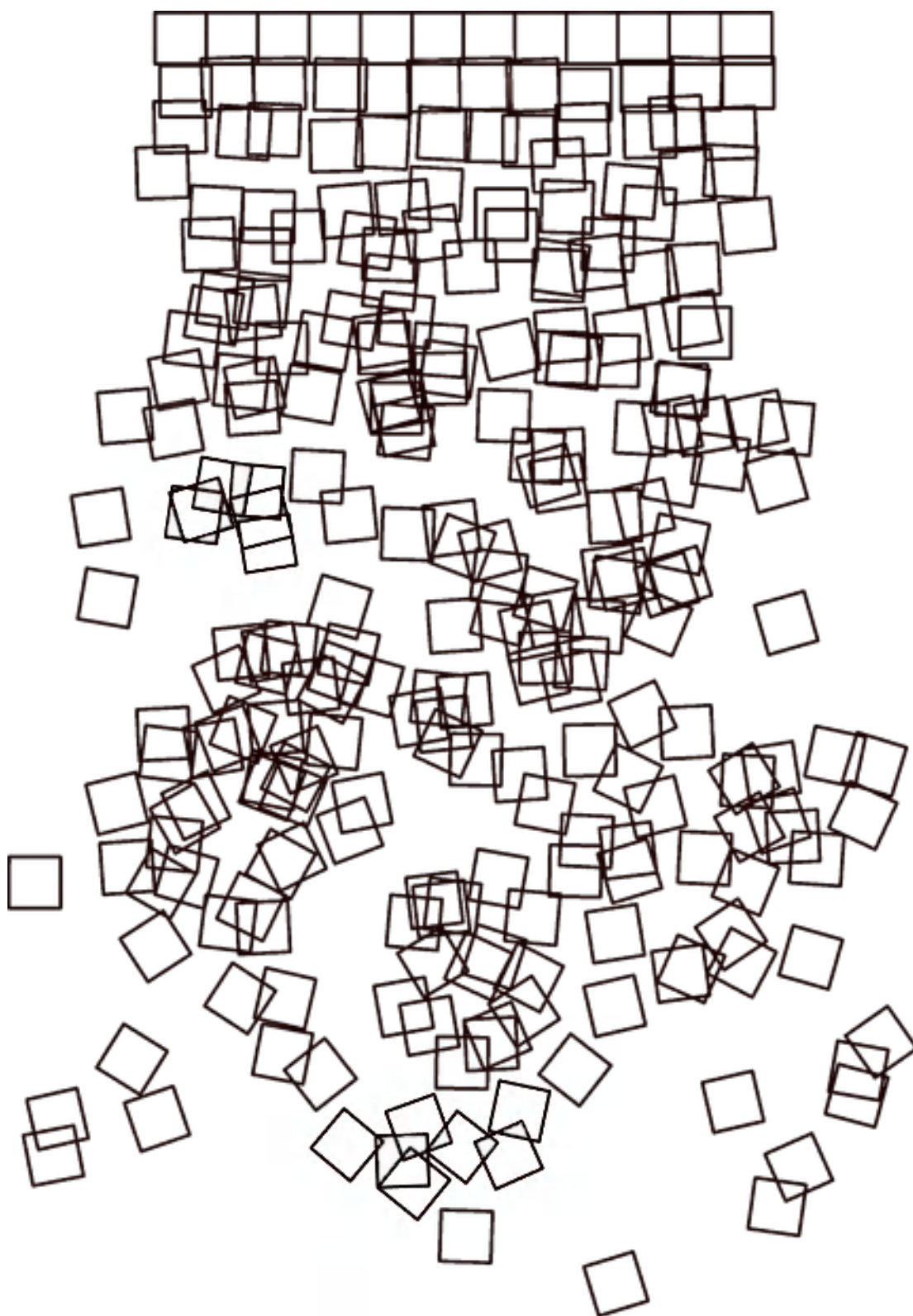


Die fünf Bilder zeigen das Ergebnis bei Beschränkung auf einzelne Kenngrößen: alle = 0, mit Drehung, mit horizontaler sowie vertikaler Verschiebung der Quadrate, sowie die Kombination dieser beiden Größen. Die beiden folgenden Bilder sind Beispiele für mein Recoding, bei dem für alle Kenngrößen gleichzeitig Werte ungleich 0 berücksichtigt sind, wobei das erste Bild dem Original sehr nahe kommt¹¹.

¹¹ Sie können dieses Recoding selber austesten: Schotter Nees original [[Programm 2](#)]



Schotter Recoded, JW 2014



Schotter Recoded, JW 2025

Das Bild *Schotter* bietet die Folie für wichtige prinzipielle Aussagen zur Computerkunst, wobei ich mich im Wesentlichen auf Frieder Nake beziehen kann. Er spricht bei digital erzeugten Bildern von der *Verdopplung der Dinge*, d.h. dass jedes digitale Kunstwerk ein Sichtbares und ein Berechenbares hat. Er benennt das sichtbare digitale Kunstwerk als die *analoge Oberfläche* (surface) und den Algorithmus bzw. das Programm als die *digitale Unterfläche* (subface) (Nake, 2016).

Als Betrachter haben wir in aller Regel keinen Zugriff auf die Unterfläche, die vom Künstler entwickelt wurde. Georg Nees allerdings erlaubt uns bei *Schotter* und vielen anderen seiner Bilder tatsächlich den Blick auf beides, auf die Unterfläche mit den Algorithmen und Programmen in seinen Büchern und auf die Oberfläche in Form der Bilder, die den Weg in viele Ausstellungen gefunden haben.

Nake spricht gern davon, dass wir *Bilder denken, nicht machen* sollten (Nake & Grabowski, 2017), denn das Machen ist bei der Computerkunst an die Maschinen übergegangen. Das Denken der Bilder impliziert, dass das einzelne Bild immer nur ein Repräsentant, eine Instanz *einer ganzen Klasse möglicher Bilder* ist, die durch die von den Urhebern dafür entwickelten Algorithmen theoretisch entstehen können. Das entsprechende Computerprogramm wird damit zur operativen Beschreibung unendlich vieler Bilder.

Das Programm *Schotter* ist durch die Verwendung der oben genannten Kenndaten, die im Grunde beliebig variiert werden können, ein typisches Beispiel einer solchen Klasse von Bildern, wie sie Nake beschrieben hat. Ganz im Sinne von Georg Nees ist es möglich und gilt es, aus dieser Vielfalt der theoretisch möglichen Ergebnisse unseres ästhetischen Laboratoriums die prägnanten Funde auszuwählen und zu dokumentieren.

Mit dem Recoding von *Schotter* bin ich, wie bereits erwähnt, weder der Erste noch der Einzige. Der generative Künstler mit dem Pseudonym *hex6c* beschreibt sein eigenes Vorgehen beim Recoding (und Remixing) von *Schotter* und anderen Vorbildern ganz analog zu dem von mir beschrittenen Weg (hex6c, 2020) wie folgt:

- Gehe aus von der Originalvorlage (bei *Schotter* ist das der Originalcode);
- Reverse Engineering des Codes (bei *Schotter* somit nicht notwendig);
- Gleiche die Ergebnisse des Codes mit der Originalvorlage ab, optimiere und passe den Code entsprechend an, um Bildversionen möglichst nahe am Original zu erhalten (bei *Schotter* durch Variation der Kenngrößen).

Wenn allerdings der Originalcode nicht vorliegt (wie es leider meistens der Fall ist), wird der Prozess etwas aufwändiger. Für mich hat sich dafür folgendes Vorgehen bewährt, das auch für die Entwicklung und Umsetzung einer eigenen Bild-idee gilt:

- Am Anfang steht immer eine genaue *Analyse einer Bildvorlage*: Welche Grundelemente sind enthalten (welche Formen, Farben und wiederkehrenden Muster und Strukturen)? Welche Operationen werden auf sie angewandt (z. B. geometrische Transformationen, Wiederholungen, Variationen)? Welche Rolle spielt der Zufall (zufällige Veränderungen von Häufigkeiten oder Formen der Bild-elemente)?
- Sobald wir daraus genau ableiten können, was wir zeichnen wollen, müssen wir beschreiben, wie die Bildelemente und die Bildstruktur erzeugt werden sollen. Dazu brauchen wir eine *algorithmische Beschreibung*. Diese beinhaltet die Festlegung der Kenngrößen wie die Anzahl und Positionierung der grafischen Elemente, sowie deren Größe, Ausrichtung und Farbe.
- Die *Programmierung* ist der letzte Schritt bei der Erstellung der Computerkunst. In dieser Phase entwickeln wir die digitale Unterfläche. Die Algorithmen sind mit Hilfe der von uns gewählten Programmiersprache so zu implementieren, dass sie von der Maschine ausgeführt werden können.

Bei meiner Analyse von Werken der frühen Computerkunst hat sich gezeigt, dass bereits einige wenige Programmierkonzepte ausreichen, die wir für die ersten Schritte in der Computerkunst benötigen. Dazu zählt die *Wiederholung* weniger bildlicher Grundelemente (bei *Schotter* reichen Quadrate), deren *Variation* sowie *systematische oder zufällige Änderungen* zentraler Kenngrößen. So lässt sich die Klasse aller von *Schotter* erzeugbaren Bilder algorithmisch beschreiben.

Der Informationstechnologe und Software-Ingenieur Jim Plaxco ist ein digitaler Künstler und kreativer Programmierer. Auch er hat das Bild *Schotter* nachprogrammiert. Plaxco hat dafür *Processing* verwendet¹², eine objektorientierte Programmiersprache mit zugehöriger integrierter Entwicklungsumgebung, die auf der Programmiersprache Java basiert und auf die Einsatzbereiche Grafik, Simulation und Animation spezialisiert ist. Sein Vorgehen hat er in einem kleinen Tutorial beschrieben (Plaxco, 2006).

¹² Sie können sein Programm herunterladen und in der Processing-Umgebung austesten: https://www.artsnova.com/Nees_Schotter.pde

```
// Georg Nees Schotter Reproduction by Jim Plaxco,
www.artsnova.com
int columns = 12;          // number of columns of squares
int rows = 22;             // number of rows of squares
int sqrsize=30;           // size of each square
float rndStep=.22;        // Rotation Increment in degrees
float randsum=0;          // Cumulative rotation value
int padding=2*sqrsize;    // margin area
float randval;            // random value for translation and
rotation
float dampen=0.45;        // soften random effect for position

void setup() {
  size((columns+4)*sqrsize,(rows+4)*sqrsize);
  background(255);        // set background color to white
  stroke(0);              // set pen color to black
  smooth();               // use line smoothing
  noFill();               // do not fill the squares with color
  rectMode(CENTER);       // use x,y value as the center of the
square
  noLoop();               // execute draw() just one time
} // end of setup()

void draw() {
  for (int y=1; y <=rows; y++){
    randsum+=(y*rndStep);  // Add to the random value
    for (int x=1; x<=columns; x++) {
      pushMatrix();
      randval=random(-randsum,randsum);
      translate( padding + (x * sqrsize) - (.5*sqrsize) +
(randval*dampen),
padding + (y * sqrsize) - (.5*sqrsize) +
(randval*dampen));
      rotate(radians(randval));
      rect(0,0,sqrsize,sqrsize);
      popMatrix();
    } // end of x loop
  } // end of y loop
} // end of draw()
```

Mit dem gezeigten Code von Plaxco erhalten Sie ein *Schotter*-Bild, das sehr dem Original ähnelt. Er schlägt auch das Austesten einiger Variationen durch Verändern einzelner Programmzeilen vor¹³.

Eine spezielle Form des Recoding hat Zellyn Hunter (2024-1) vorgenommen. Seine *Schotter*-Version hat er mit der Programmiersprache Python implementiert. Er orientiert sich mit Variablen- und Prozedurnamen ganz nah am Originalcode.

¹³ Jörg Kantel (der *Schockwellenreiter*) hat von diesem Programm eine Fassung mit Processing.py - der Python-Fassung von Processing - erstellt und ebenfalls den Code zur Verfügung gestellt: <http://blog.schockwellenreiter.de/2019/04/2019042501.html>

```

# SCHOTTER
R = 5 * 1.4142
HALFPI = 3.14159 * 0.5
PI4T = HALFPI * 0.5
I = 0
P = Q = PSI = 0.0

def QUAD():
    global I, JE1, JA1, JE2, JA2

    JE1 = 5*I/264
    JA1 = -JE1
    JE2 = PI4T * (1+I/264)
    JA2 = PI4T * (1-I/264)
    P1 = P+5+J1()
    Q1 = Q+5+J1()
    PSI = J2()

    LEER(P1+R*COS(PSI), Q1+R*SIN(PSI))
    for S in (1,2,3,4):
        PSI = PSI+HALFPI
        LINE (P1+R*COS(PSI), Q1+R*SIN(PSI))
    I += 1

def SERIE(QUER, HOCH, XMAL, YMAL, FIGUR):
    global P, Q

    P = -QUER * XMAL * 0.5
    Q = YANF = -HOCH * YMAL * 0.5

    for COUNTX in range(1, XMAL+1):
        Q = YANF
        for COUNTY in range(1, YMAL+1):
            FIGUR()
            Q = Q + HOCH
            P = P + QUER
        LEER(-148.0, -105.0)
        CLOSE()

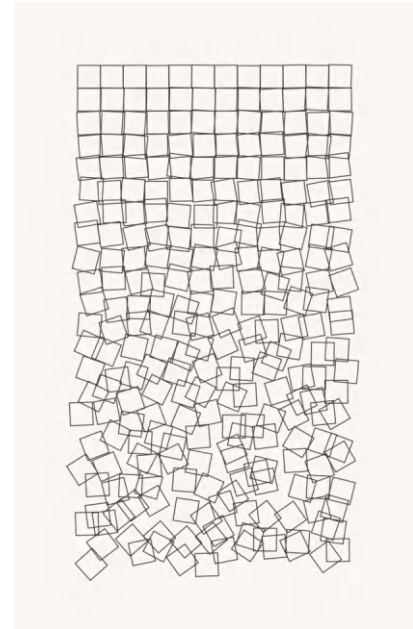
SERIE(10.0, 10.0, 22, 12, QUAD)
# /SCHOTTER

```

Zellyn Hunter gibt sich allerdings nicht damit zufrieden, das Prinzip *Schotter* zu Recoden und dem Original möglichst ähnliche Bilder zu erzeugen. Er möchte das Original bis ins Detail korrekt wiedergeben, also alle Positionen und Drehungen der Quadrate in den 22 Reihen.

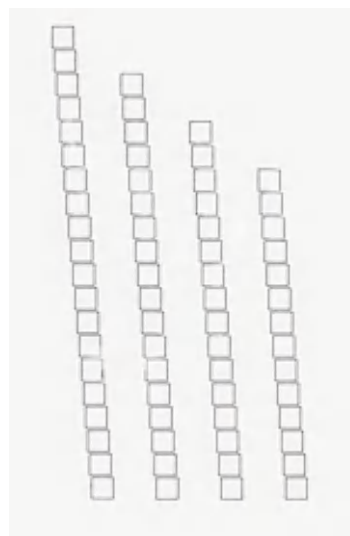
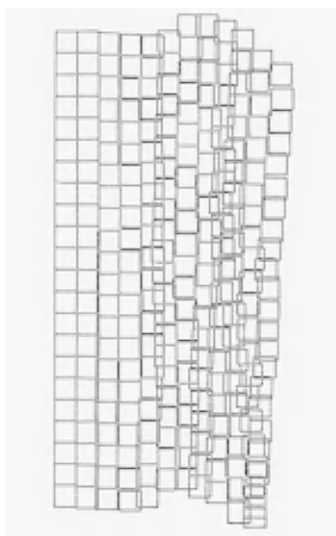
Dazu bedarf es allerdings der Nutzung des gleichen Zufallszahlengenerators, den Nees eingesetzt hat. Hunter ist in der Dissertation von Nees fündig geworden (Nees, 1969, S. 113 ff.).

Mit diesem Generator und dem Auffinden der Startwerte für die x- und y-Verschiebung sowie der Rotation ist es ihm gelungen eines der Originale 1:1 zu reproduzieren (Hunter, 2024-2; er bezieht sich auf die Version, die sich in der Sammlung des Victoria and Albert Museum befindet, vgl. S. 12).



*Hunter: 1:1 Recoding von Schotter
(Version des V&A Museum)*

Der Medienwissenschaftler Martin Zeilinger befasst sich mit Fragen der Autorenschaft, Kreativität, Originalität und Eigentum in neuen technologischen Kontexten. In seinem Projekt *Pattern Recognition* kombiniert er einen kritischen Ansatz mit einem künstlerischen Ansatz (Zeilinger, 2018). Am Beispiel von *Schotter* demonstriert er durch die detaillierte Analyse des Originalwerks, dem Neuschreiben



Zeilinger: Recoding Schotter Iteration_7488 (links); Iteration_0041 (Mitte); Iteration_7780 (rechts), 2017

des Codes und der Reproduktion unter Verwendung von mit den Originalen vergleichbaren technischen Geräten, welche Aspekte der künstlerischen Zusammenarbeit zwischen Mensch und Computer bei der Generativen und Algorithmischen Kunst zugrunde liegen.

Durch die Verbindung mit sozio-juristischen Fragen spricht Zeilinger ein grundsätzliches Problem beim Recoding und Remixing an, das sich aktuell bei der Generativen KI in neuer Schärfe zeigt. Entstehen dabei nicht lediglich Imitate, Nachbildungen, also Wiederholungen künstlerischer Werke? Im Grunde stellt sich unter dem Aspekt des Plagiarismus die Frage neu, mit der auch die frühen Computerkünstler konfrontiert wurden, nämlich was denn Kunst an ihren Werken sei.

Schon Schneeberger (2013) diskutierte am Beispiel des ReCode Project¹⁴ die Grenzen zwischen dem konservatorischen Bemühen, digitale Kunst zu bewahren beziehungsweise zugänglich zu machen, und Plagiarismus. Sein kurzer Essay zu diesem Thema sollte Pflichtlektüre sein, wenn aus den Programmierübungen im stillen Kämmerlein Produkte werden, die den Weg in den öffentlichen Raum finden sollen!

¹⁴ Das ReCode-Projekt ist ein Community-getragenes Projekt zur Bewahrung der Computerkunst durch deren Übersetzung in eine moderne Programmiersprache (Processing): <http://recodeproject.com/>

Remixing Schotter

Das Recoding ist oft einfach - selbst ohne Verwendung des Original-Codes - wenn die Bildstrukturen relativ einfach nachvollziehbar sind. Dem Recoding folgt aber nun das *Remixing*! Darunter wird das Überarbeiten und Kombinieren kreativer Artefakte verstanden, üblicherweise in der Form von Musik, Video oder anderen interaktiven Medien, hier analog von mir bezogen auf bildliche Vorlagen. Das Recoding zielt auf die weitgehend ähnliche Reproduktion der jeweiligen Vorlage ab (im Extremfall sogar mit dem Anspruch der identischen Wiedergabe wie bei Zellyn Hunter). Beim Remixing dagegen dient das Original als Ideengeber für kleinere oder auch größere bildliche Veränderungen.

In unserem Kontext bedeutet Remixing also, dass die beim Recoding erzeugten Bilder und Ausgangselemente neu kombiniert, abgewandelt und erweitert werden können und damit neue Werke entstehen. Das kann von behutsamen Veränderungen nah am Original bis zur freien Kombination bekannter Elemente in neuen Konstellationen reichen. Die können bereits durch kleine Ergänzungen oder Veränderungen des Originalcodes erreicht werden. Wird aber die bildliche Struktur verändert, kann das eine Umprogrammierung oder sogar Neuprogrammierung erfordern.

Bei der Computerkunst können die verwendeten Artefakte deshalb zwei unterschiedlichen Ebenen entstammen:

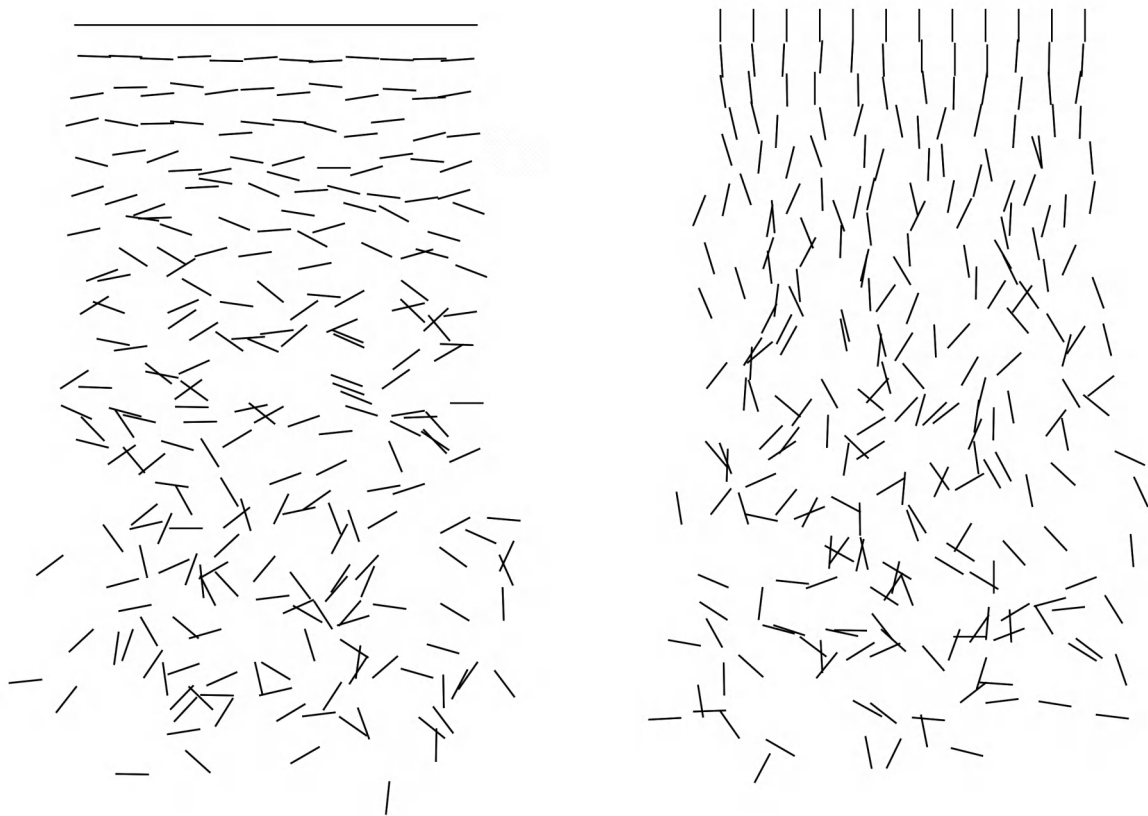
- Es können zum einen die grafischen Ergebnisse selbst sein bzw. einzelne daraus entnommene Elemente und Strukturen, die in neuen Zusammenhängen und Zusammenstellungen auftauchen können.
- Es können zum anderen die Algorithmen bzw. die Programme sein, mit denen die ästhetischen Objekte erzeugt wurden, aus denen dann einzelne Prozeduren bzw. Code-Schnipsel entnommen werden.

Eine Grundhaltung beim Remixing sollte sein, dass sowohl die Quellen erkennbar bleiben und benannt werden, als auch ein transparenter Umgang mit diesen Quellen (Stalder, 2009).

Im Folgenden möchte ich etliche Beispiele für ein so verstandenes Remixing vorstellen. Sie stammen von mir und anderen, die am Beispiel *Schotter* in diesem Sinne weiter gearbeitet haben. Die Ergebnisse ergeben sich teils durch leichte Veränderungen gegenüber dem Original, andere sind davon inspirierte Neufassungen.

Ich habe 2015 in einer Ausstellung, damals anlässlich 50 Jahren Computerkunst, neben dem Recoding von *Schotter* mehrere Beispiele für das Remixing gezeigt.

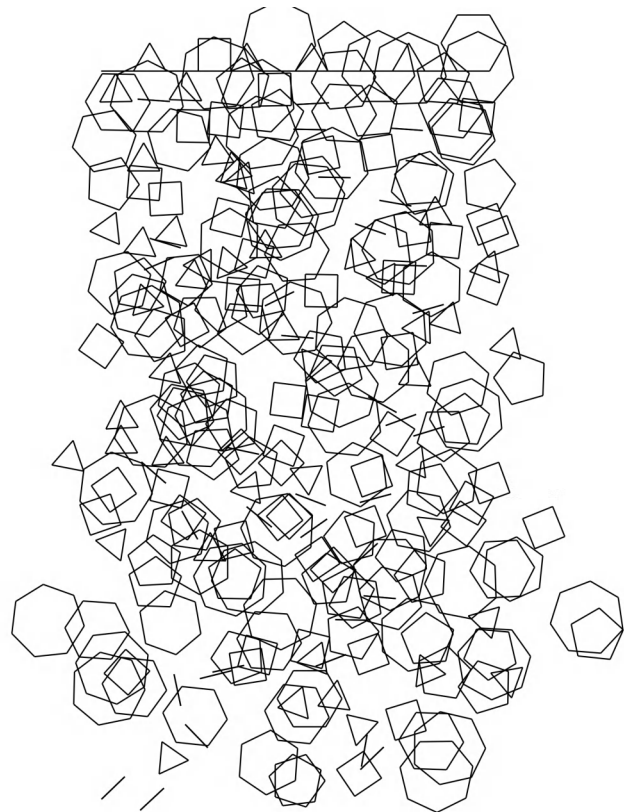
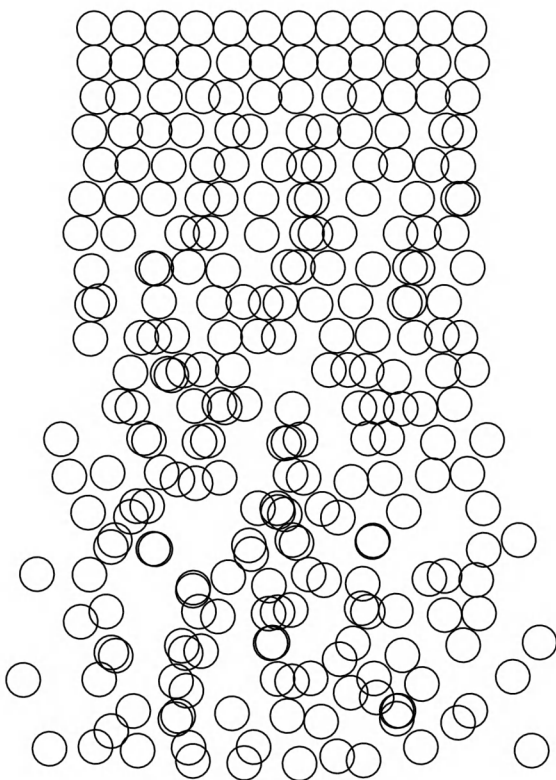
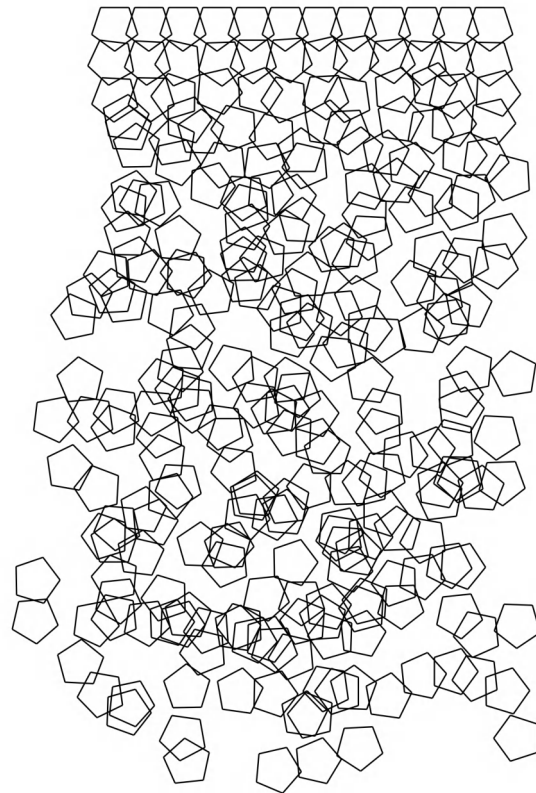
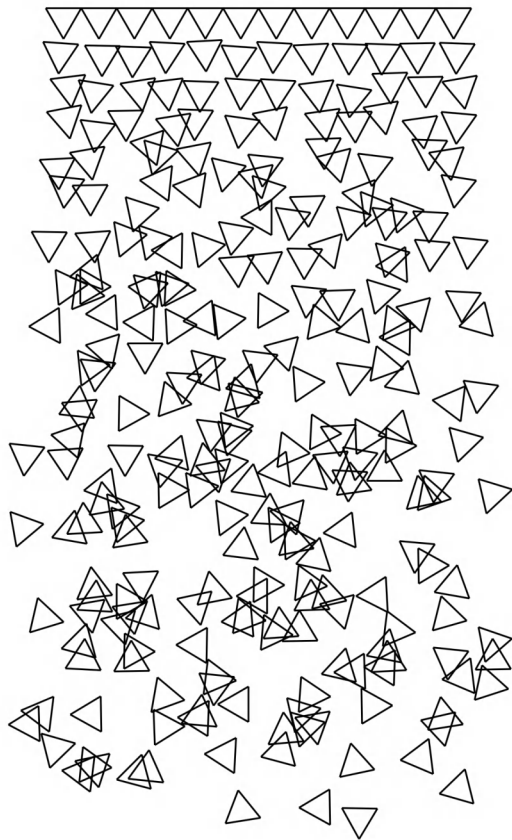
Eine erste Serie zeigte - abweichend von den Quadraten als Grundelement mit vier Eckpunkten - Polygone mit beliebig vielen Eckpunkten. Bei zwei Eckpunkten ergeben sich daraus Linien, die in der Ausgangslage horizontal oder vertikal orientiert sein können.



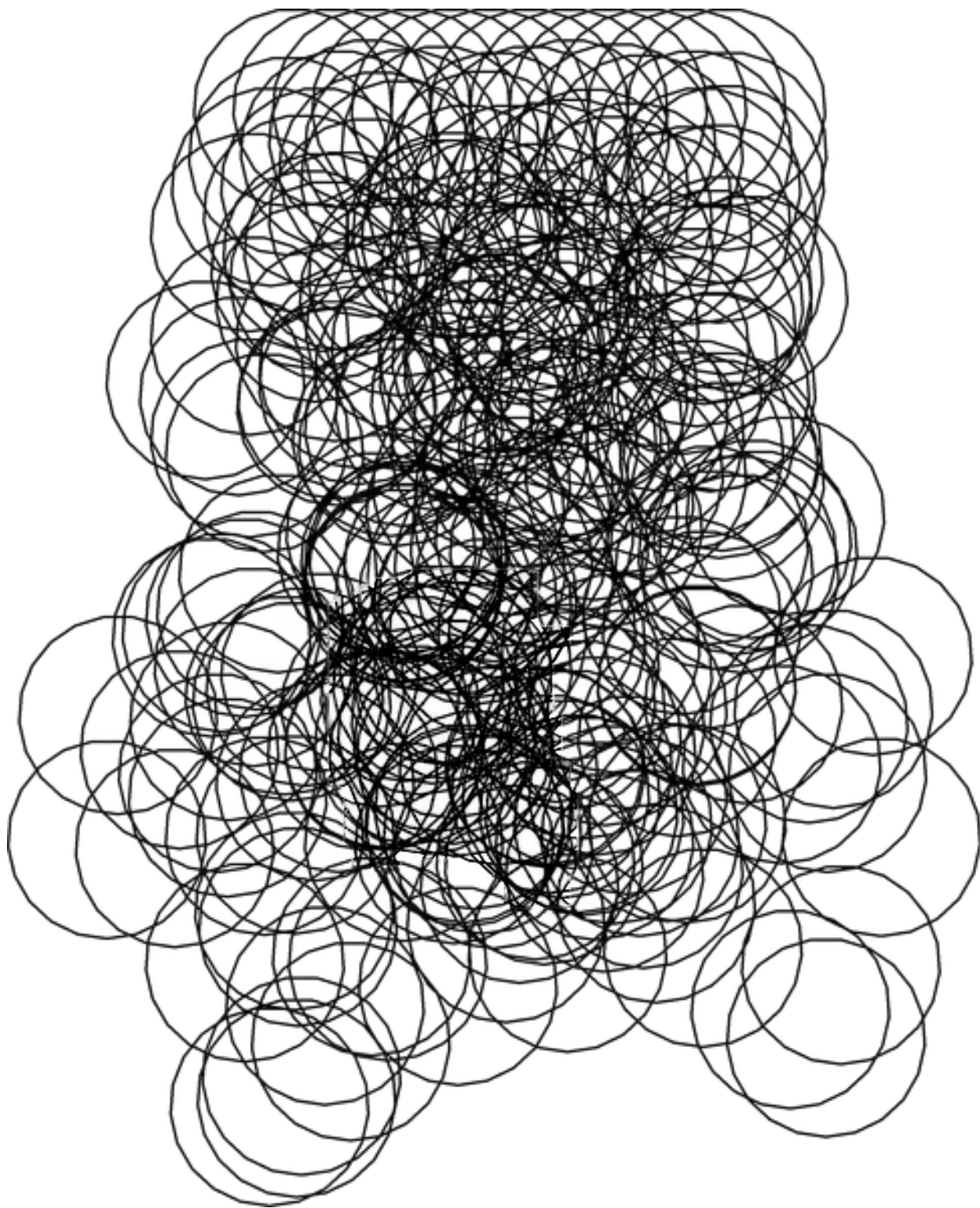
Schotter Remixed, Zwei Eckpunkte, horizontal (links) und vertikal (rechts) ausgerichtet, JW 2015

Alle anderen Polygone mit drei oder mehr Eckpunkten verhalten sich im Prinzip wie die Quadrate in *Schotter*. Bei sehr vielen (>20) Eckpunkten sind es Kreise, bei denen der Faktor Drehung dann keine Rolle mehr spielt.

Solche Varianten sind programmtechnisch sehr leicht umzusetzen. In meiner oben gezeigten Snap!-Prozedur **Schotter** ist lediglich der Aufruf der Prozedur **quadrat** durch eine Prozedur **polygon** zu ersetzen, der jeweils die gewünschte Eckenzahl zu übergeben ist. Die ursprüngliche *Schotter*-Fassung ist so automatisch mit enthalten, wenn die Prozedur **polygon** mit vier Eckpunkten aufgerufen wird (so auch in meiner Version *Schotter total interaktiv*, siehe S. 48).

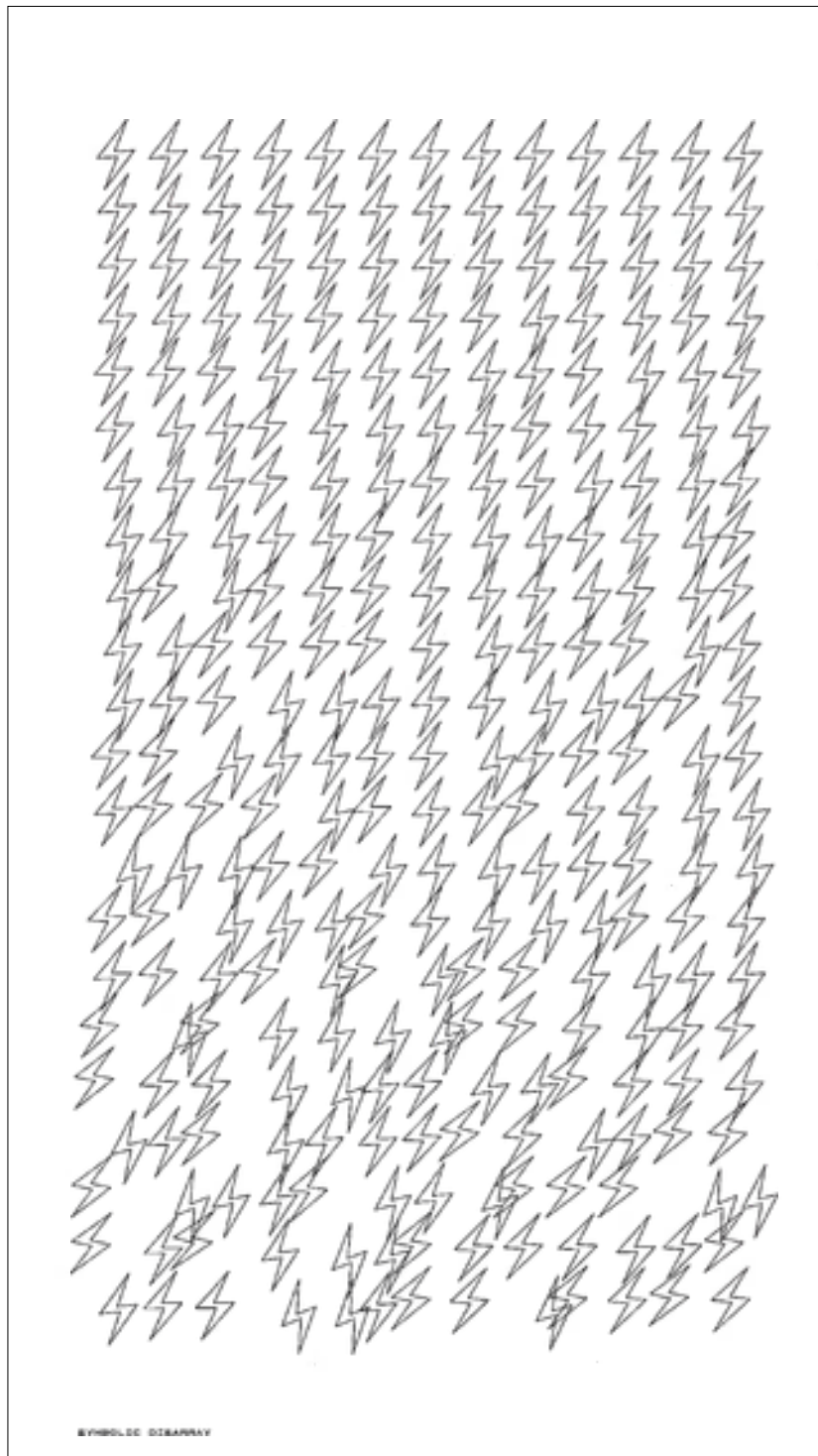


*Schotter Remixed, Dreiecke (oben links), Fünfecke (oben rechts), 20-Ecke (unten links)
zufällige Eckenzahl (unten rechts), JW 2015 - 2025*



Schotter Remixed, 20-Ecke, JW 2015

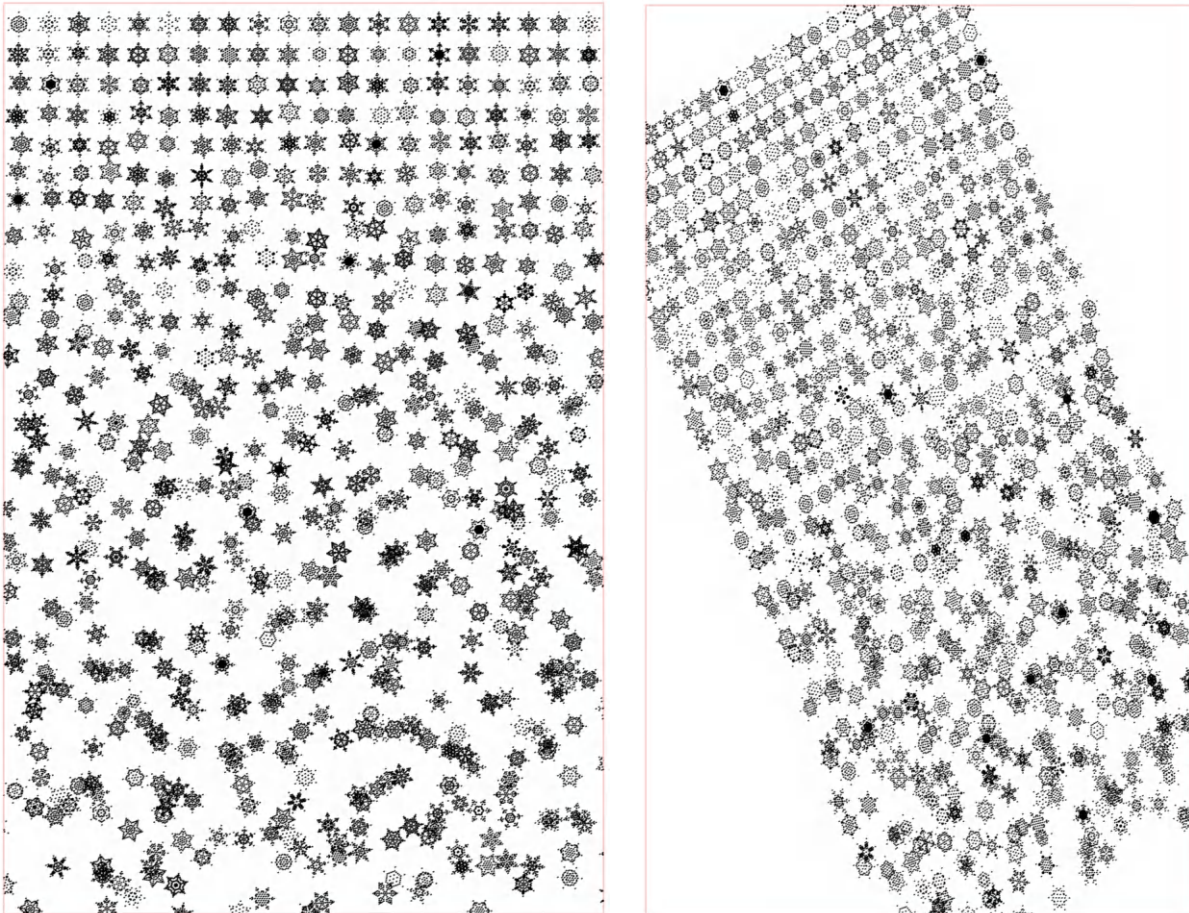
Dieses Vorgehens kann nochmals dadurch stark verallgemeinert werden, dass die dargestellten Bildelemente ganz neue Formen annehmen. Bei Tobias Toft sind es zum Beispiel stilisierte Blitze (Toft, 2014)¹⁵.



Tobias Toft: Symbolic Disarray, 2014

¹⁵ Sie können sein Programm herunterladen und austesten (dazu werden in diesem Fall etwas Processing-Kenntnisse vorausgesetzt): <https://github.com/tobiastoft/SymbolicDisarray>

Der Grafiker Daniel Catt beschäftigt sich mit wichtigen Aspekten der Generativen Kunst. Bei seinem Remixing von *Schotter* werden die Quadrate durch Schneeflocken ersetzt. Sein Programm *Snowflakes* ist allerdings mehr als ein reines Remixing von *Schotter*. Es ist eine Arbeitsumgebung, in der Sie eine Vielzahl von Kenngrößen sehr einfach einstellen und die Änderungen am Bild live verfolgen können¹⁶. Experimentieren lohnt sich!



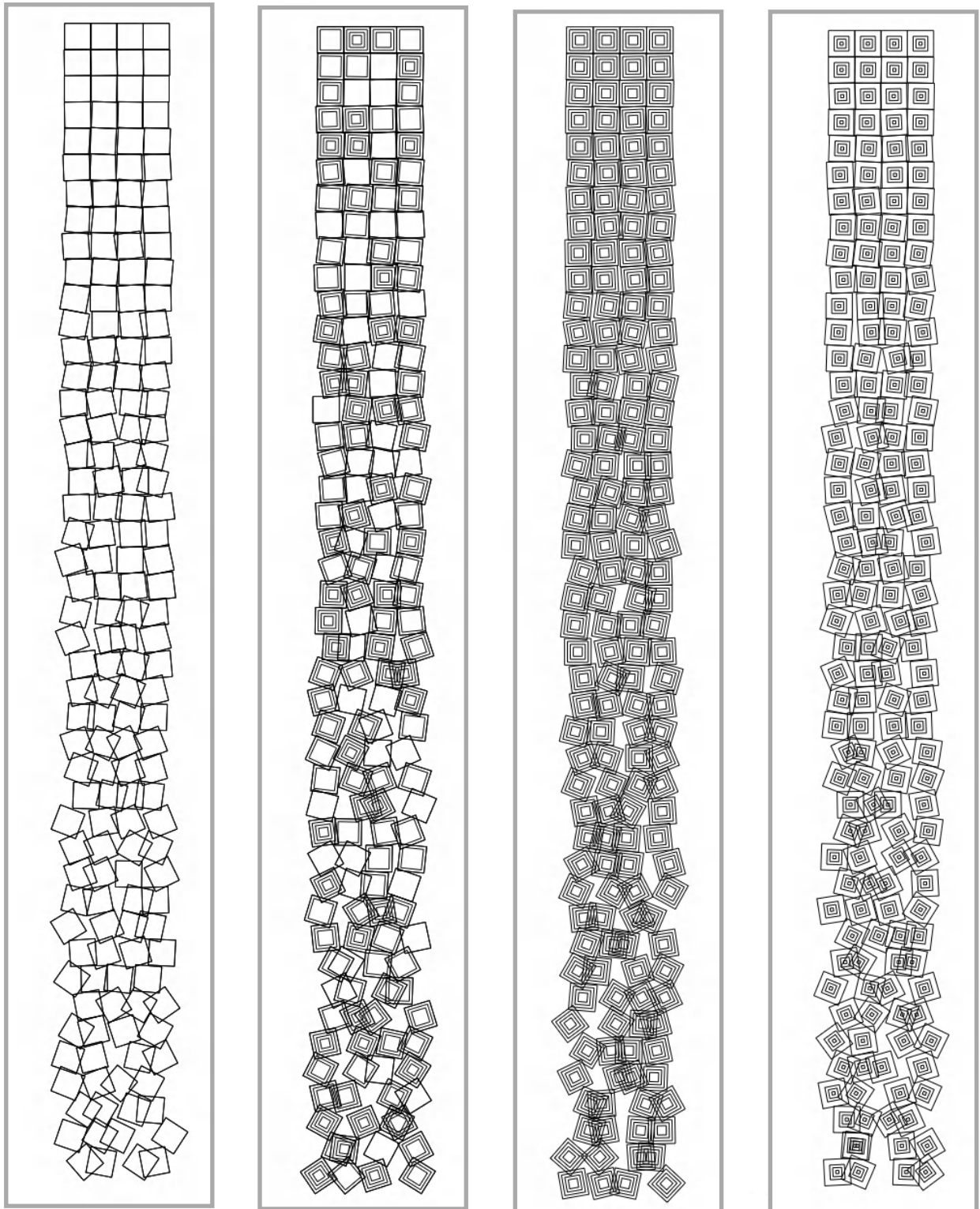
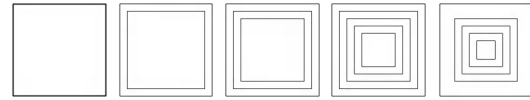
Daniel Catt: Schotter Snowflakes (2021), Voreinstellung (links), x-y-z-Rotation (rechts), JW 2025

Für eine Ausstellung *Malen mit Maschinen*¹⁷ habe ich die Grundidee von Schotter dahingehend abgewandelt, dass geschachtelte Quadrate in einer langen Bahn durchgeschüttelt werden. Entstanden sind damit Polygonfahnen, die vom Boden bis in den Deckenbereich des Ausstellungsraumes reichten. Verwendet habe ich

¹⁶ Seine Umgebung läuft ohne Installation direkt im Webbrowser: <https://revdancatt.com/penplotter/036-Snowflakes/>

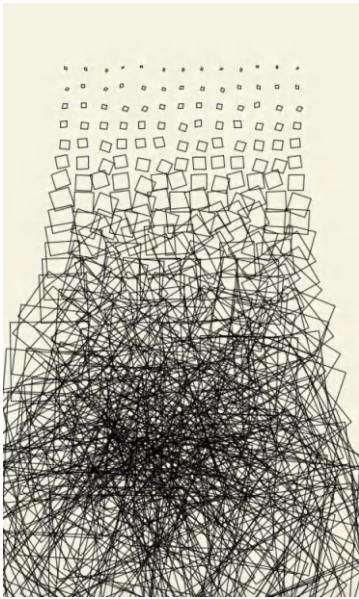
¹⁷ Eine Dokumentation der Ausstellung findet sich hier <http://digitalart.joachim-wedekind.de/malen-mit-maschinen-2023/>

dabei fünf grafische Grundelemente, die jeweils in einer 4x4 Matrix mit gelenkten Zufallseinflüssen gezeichnet werden.

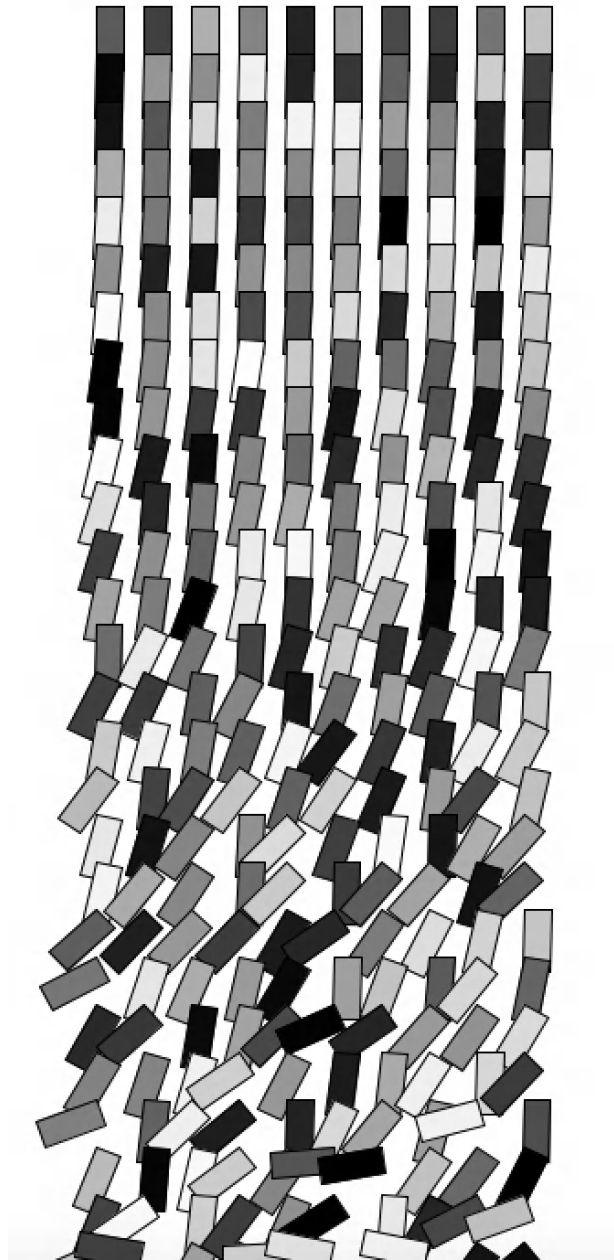
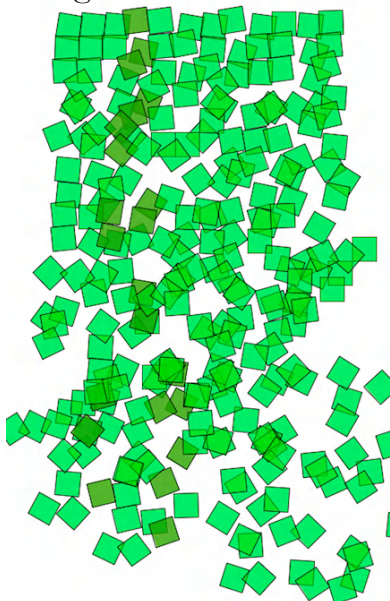


Polygonfelder 4x4 I-IV, JW 2023

An der Kunsthochschule Sarah Lawrence College hat Angela Ferraiolo mehrfach den Kurs *Art from Code*¹⁸ angeboten. Ein wesentliches Element ist dabei das Re-coding & Remixing von Werken der frühen Computerkunst. Also ist es nicht verwunderlich, dass im Kurs 2018 auch *Schotter* von Georg Nees dazu herangezogen wurde mit der Aufgabe *A response to Georg Nees*. Das Original war wohl inspirierend, wie die sehr unterschiedlichen Ergebnisse zeigen. Beim Remixing wurden neue Formen verwendet und es wurde oft mit Farben gearbeitet.

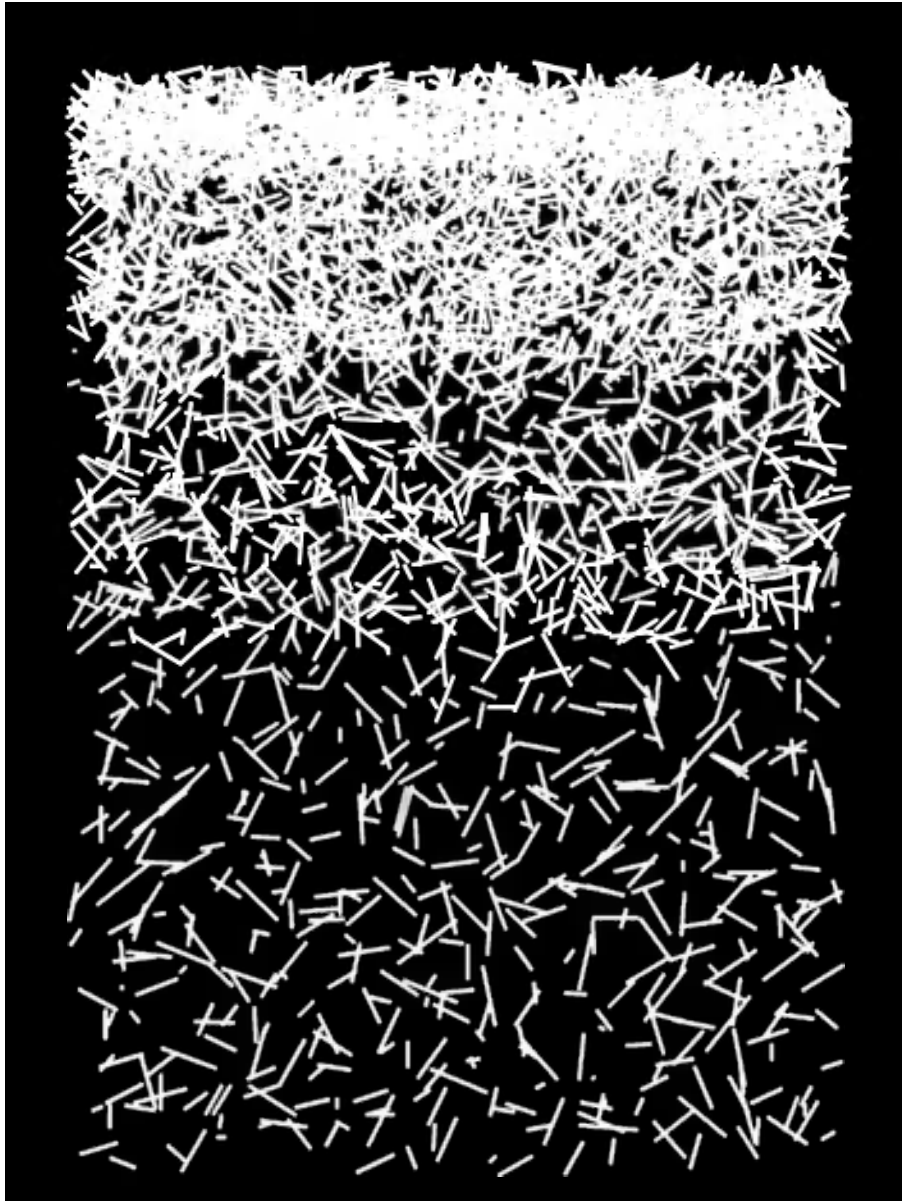


Emma Hoffmann: A response to Georg Nees, 2018



Vivian Zhou (links); Yaman Wang (rechts): A response to Georg Nees, 2018

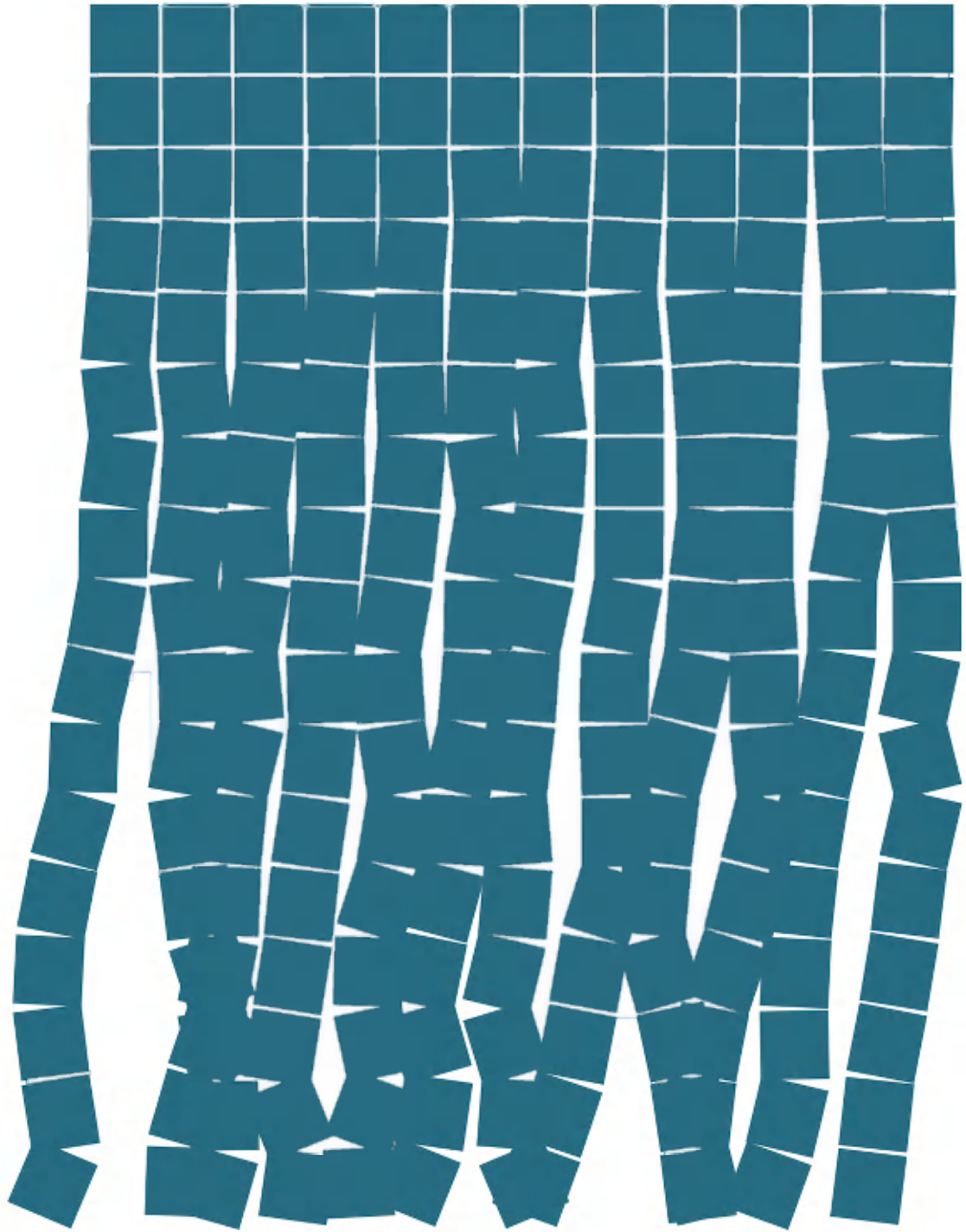
¹⁸ Eine Dokumentation der Kursergebnisse findet sich unter <https://astoryisnotatree.net/?cat=225>



Jordan Hiedeman: A response to Georg Nees, 2018

Die Arbeit von Hiedeman ähnelt meinem Remix von *Schotter* mit Polygonen mit zwei Eckpunkten, hier mit variabler Ausrichtung und Verdichtung zum Bildrand hin.

Allein die Verwendung von Farbe (wie bei Zhou oder Hiedemann) ergibt bereits eine deutliche Veränderung des Bildcharakters, selbst bei sonst vergleichbarer Bildstruktur. Das zeigt sich auch bei meinen eigenen Farbversionen: Die Version *Vorhang* verwendet einheitlich gefärbte Quadrate. Bei *Loses Mauerwerk* wird die Farbe mit der vertikalen Position immer transparenter.

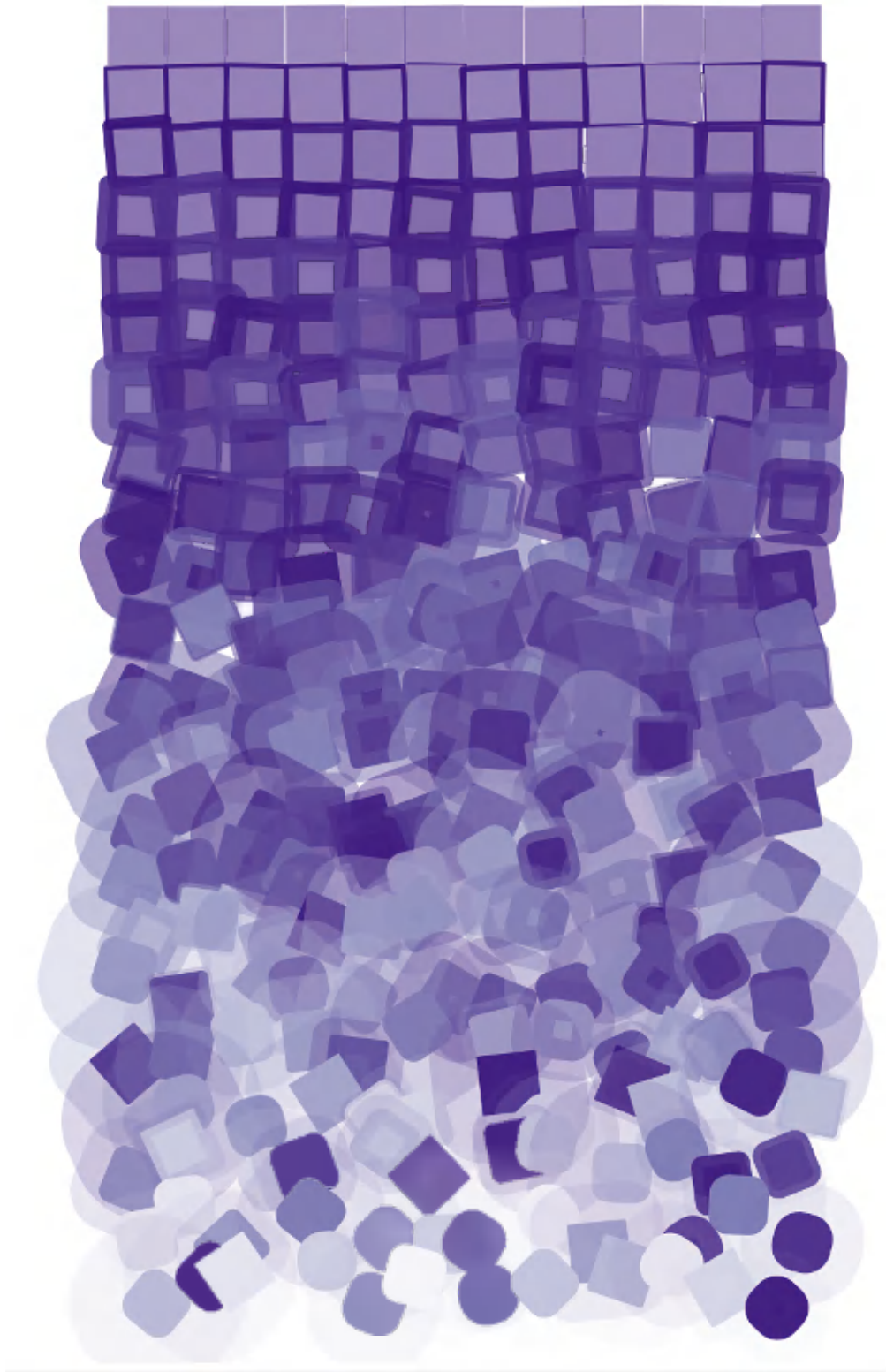


Remixing Schotter: Vorhang, JW 2019



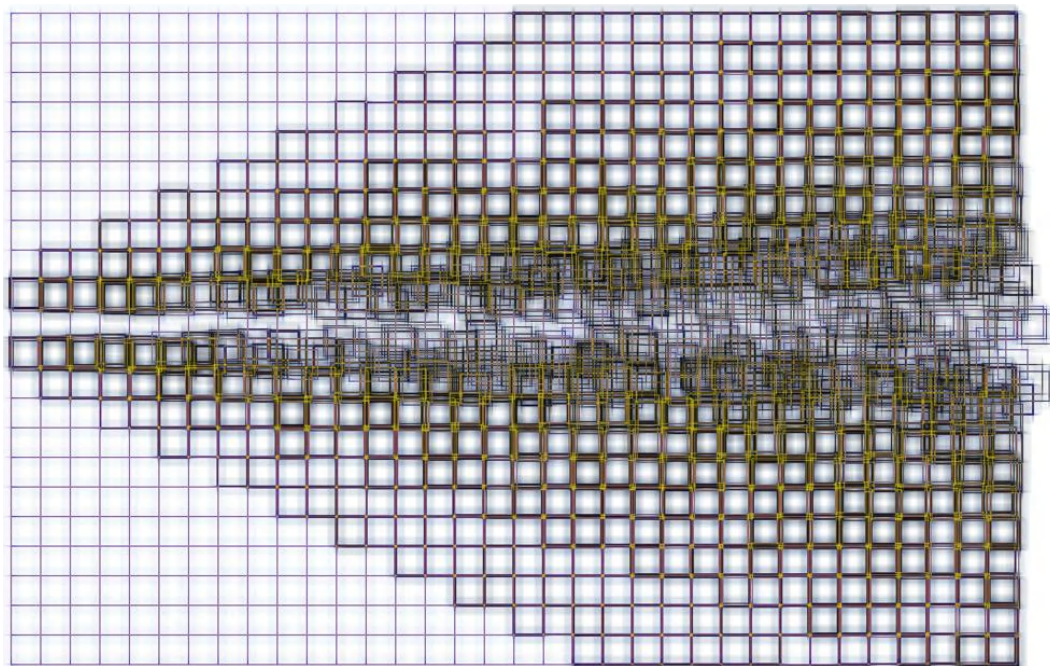
Remixing Schotter: Loses Manerwerk, JW 2022

Auch der Datenvisualisierer Ilyá Belsky arbeitet mit Farbe. Bei ihm sind Quadrate und die überlagernden Polygone mit zunehmender Transparenz eingefärbt:

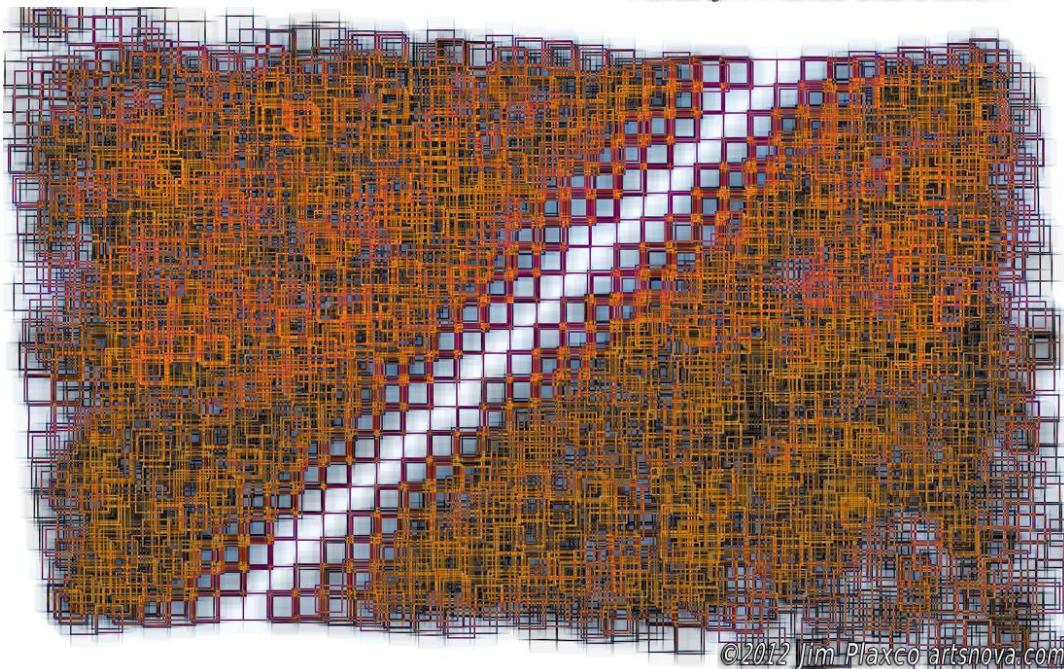


Ilyá Belsky: Remixing Schotter, 2019

Der schon erwähnte Jim Plaxco hat eine ganze Serie *Cubic Disarray* entwickelt, zu der er von *Schotter* angeregt wurde. Seine Bilder sind seltene Beispiele des Remixing, bei denen die übliche vertikale Bildausrichtung durch eine horizontale Ausrichtung ersetzt wird. Durch Farbe und die Anordnung mit Überlagerungen ist der Bezug zu Schotter nicht mehr direkt erkennbar.



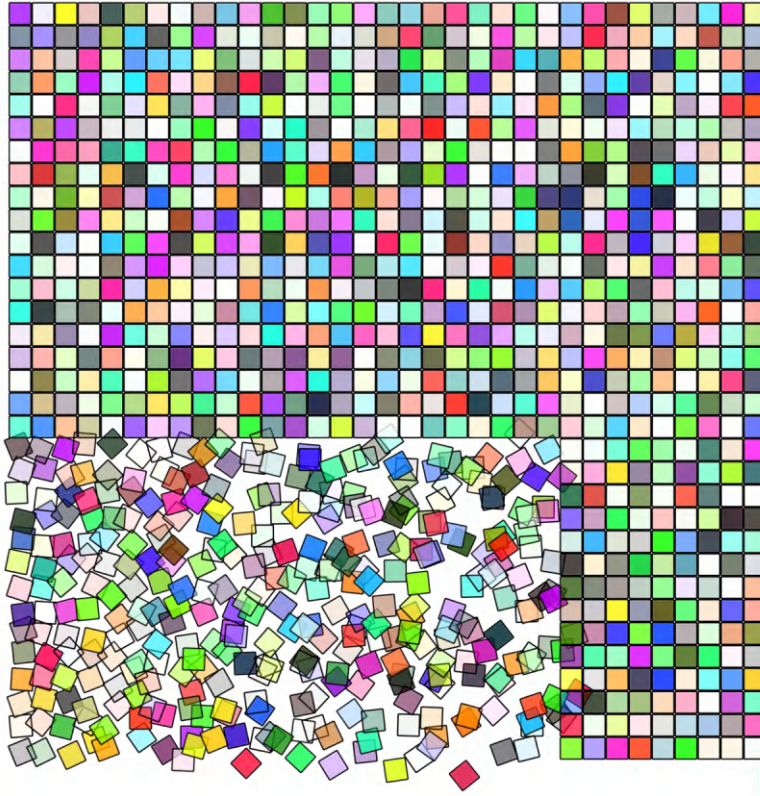
©2011 Jim Plaxco artsnova.com



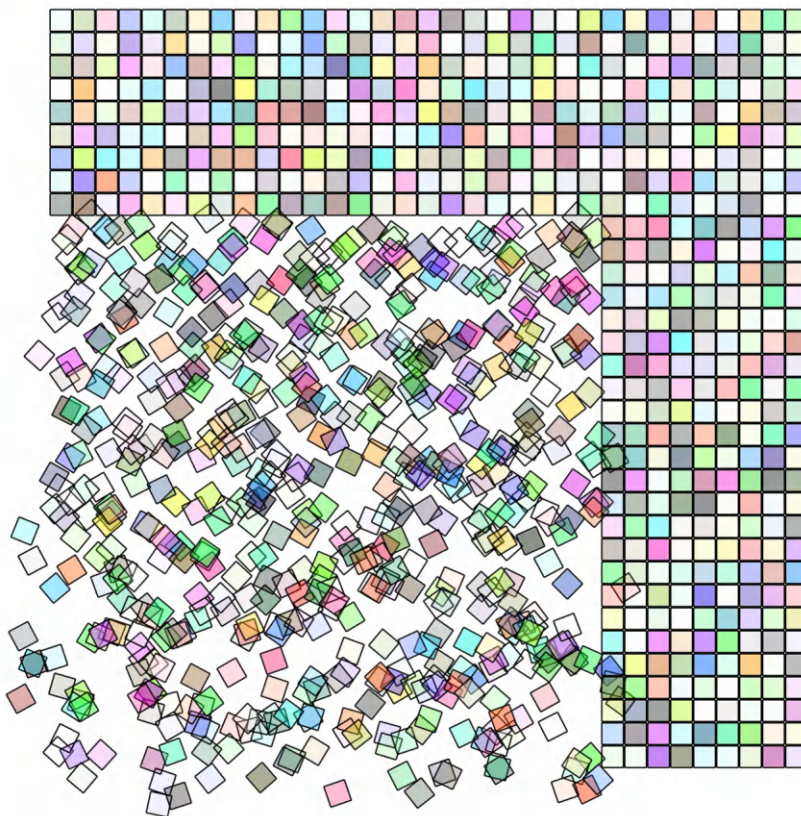
©2012 Jim Plaxco artsnova.com

Jim Plaxco: Cubic Disarray - Turbulence (oben), 2011; Bisection (unten), 2012

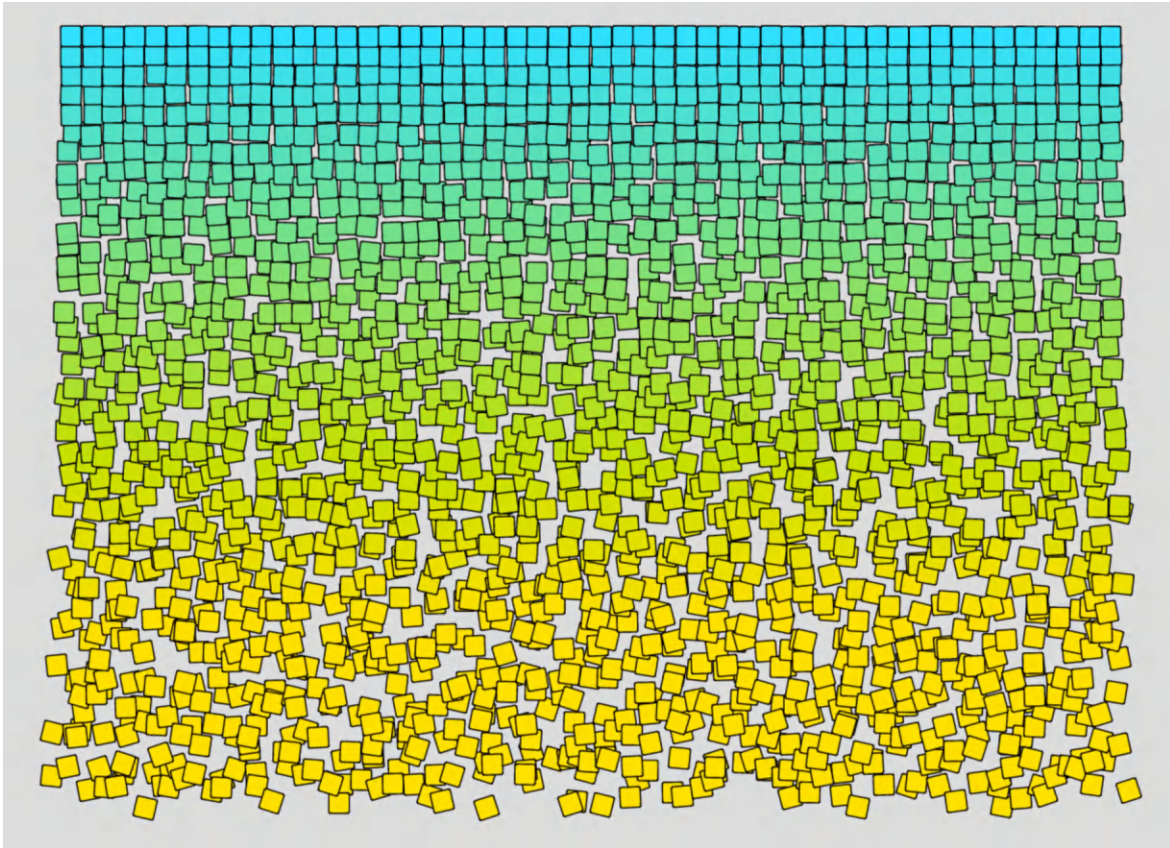
Im folgenden Remixing *Mauerfall I&II* habe ich - neben der zufälligen Einfärbung der Quadrate - die Zufallsänderungen jeweils auf einen Teilbereich der „Mauer“ eingegrenzt. Die Zufallsänderung betrifft auch die Transparenz der Farbquadrate.



*Remixing Schotter: Mauerfall I,
JW 2017*



*Remixing Schotter: Mauerfall II,
JW 2017*

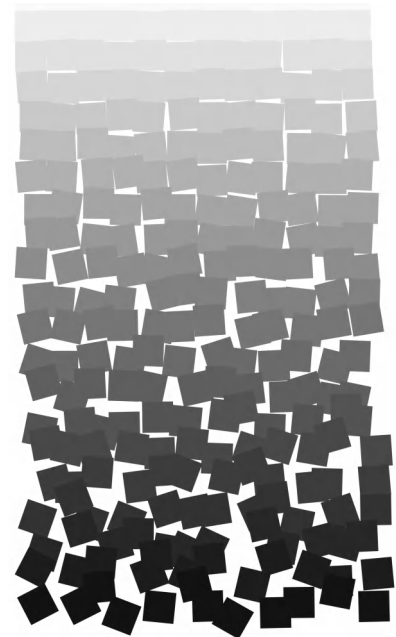


Remixing Schotter: Breitwandformat, JW 2017

Bei meinem Bild *Schotter Breitwandformat* betrifft das Remixing das Bildformat und den Einsatz eines Farbverlaufs.

Der Informatikprofessor und Generative Künstler Massimo Franceschet (Pseudonym hex6c) hat ein Projekt *Generative Art Recoded* (GARP) durchgeführt (hex6c, 2020), mit dem er ausgewählte Werke der frühen Computerkunst durch Nachprogrammieren erhalten möchte.

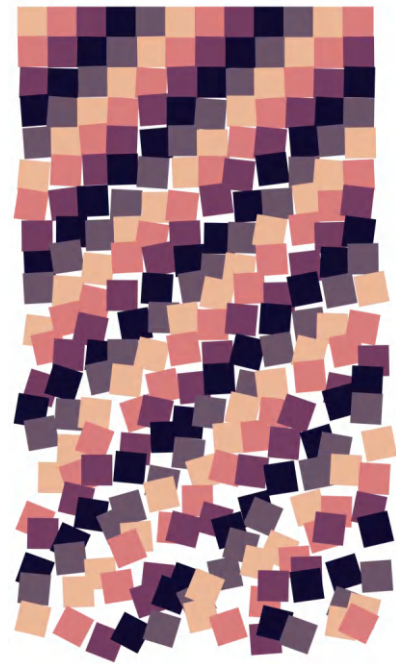
Er wendet sein typisches Vorgehen dabei (siehe Kapitel *Recoding*) neben *Schotter* auch auf *Computer Composition With Lines* von Michael Noll, *Hommage à Paul Klee* von Frieder Nake und *(Dés)Ordres* von Vera Molnar an. Bei *Take I* verwendet er farbige (schwarze) Quadrate und eine von oben nach unten abnehmende Transparenz (ganz ähnlich dem Remixing



hex6c: Schotter - Take I, 2020

Loses Mauerwerk von mir und dem *Remixing Schotter* von Ilyá Belsky. Bei *Take II* verwendet er fünf Farben ohne Änderung der Transparenz.

Bei den Bildern *Take III* und *Take IV* führt hex6c als Ersatz für die Quadrate (oder Polygone) Buchstabenfolgen und Ziffernfolgen ein. Dieser Austausch der Bildelemente - ebenso wie bei den oben gezeigten Bildern von Tobias Toft oder Daniel Catt - orientiert sich in der Bildstruktur zwar immer noch am Vorbild *Schotter*, erlaubt aber ganz neue Bildkompositionen.



hex6c: Schotter - Take II, 2020



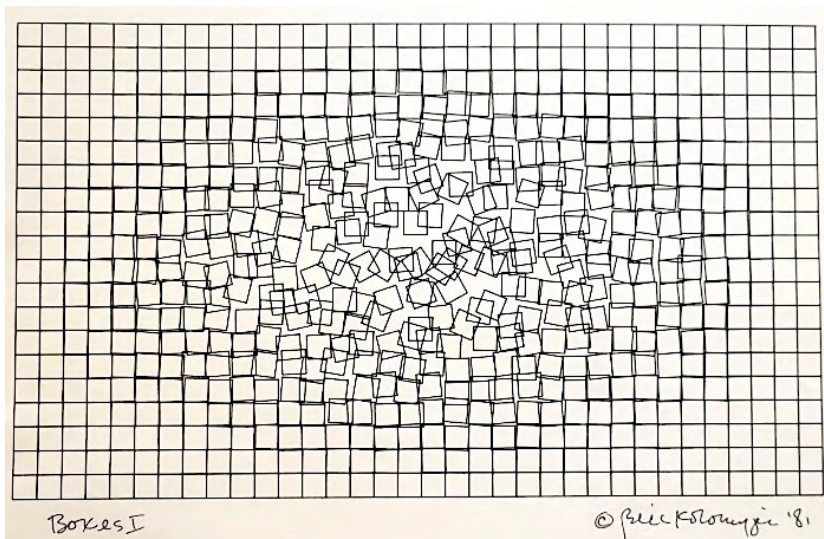
hex6c: Schotter - Take III, 2020



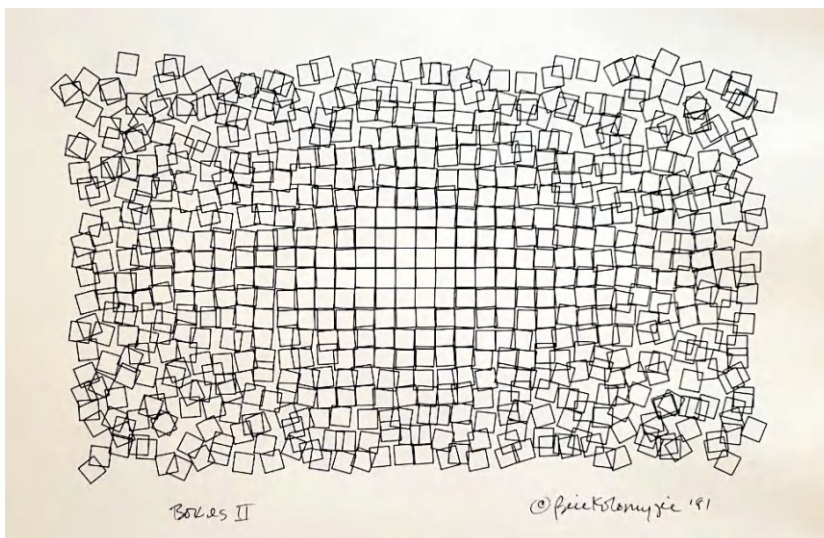
Take IV, 2020

Ein früher und heute noch sehr aktiver Vertreter der Generativen Kunst ist William J. Kolomyjec. Zu seiner Bildserie *Boxes I & II* ist er von Schotter angeregt worden. Diese Bilder sind schon 1973 entstanden und können als erste Beispiele für ein Remixing der frühen Computerkunst gelten. Bill Kolomyjec ist damit wahrscheinlich einer der ersten, der sich explizit auf diese Vorbilder bezog. Er begründet den Bezug dazu wie folgt (Kolomyjec, 1975):

„It was not uncommon for these early computer image makers to use the computer to duplicate a basic theme by a traditional artist. A classic study is by A. Michael Noll, Computer Compositions with Lines 1965, after Piet Mondrian's Composition with Lines. Boxes in turn is my interpretation of a work by George Nees entitled Gravel Stones (Schotter). I feel that it is appropriate to study works of other artists, especially traditional artists and those who established computer art in the fine art realm.“



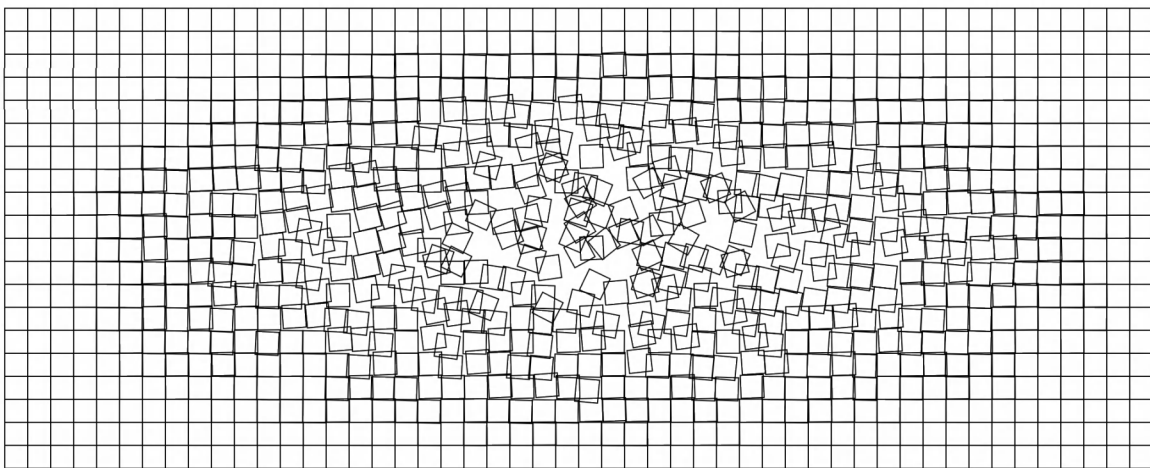
Kolomyjec: Boxes I. 1981



Kolomyjec: Boxes II. 1981

Die Grundstruktur bei *Boxes* ist ein regelmäßiges Gitter aus quadratischen Kacheln. Darin besteht die Nähe zur Ausgangslage bei *Schotter*. Im Fall von *Boxes I* nehmen die zufälligen Drehungen und die Verschiebungen aber zu, je näher die einzelnen Quadrate am Zentrum des Gitters liegen. Im Fall von *Boxes II* dagegen nehmen sie zu, je weiter die einzelnen Quadrate vom Zentrum des Gitters entfernt sind¹⁹.

Interessanterweise gibt es bereits das Recoding dieses Remixing. Als Teil des *Re-Code Project* hat Hendrik Kleper sich verschiedene Klassiker der Computerkunst zum Vorbild genommen (John Whitney, Vera Molnar, Roger Coart u.a.). Sein Recoding von *Boxes I* erlaubt die Reproduktion von Bildern nahe am Original²⁰. Auch die neueren Bilder im quadratischen Format lassen sich so leicht erzeugen.



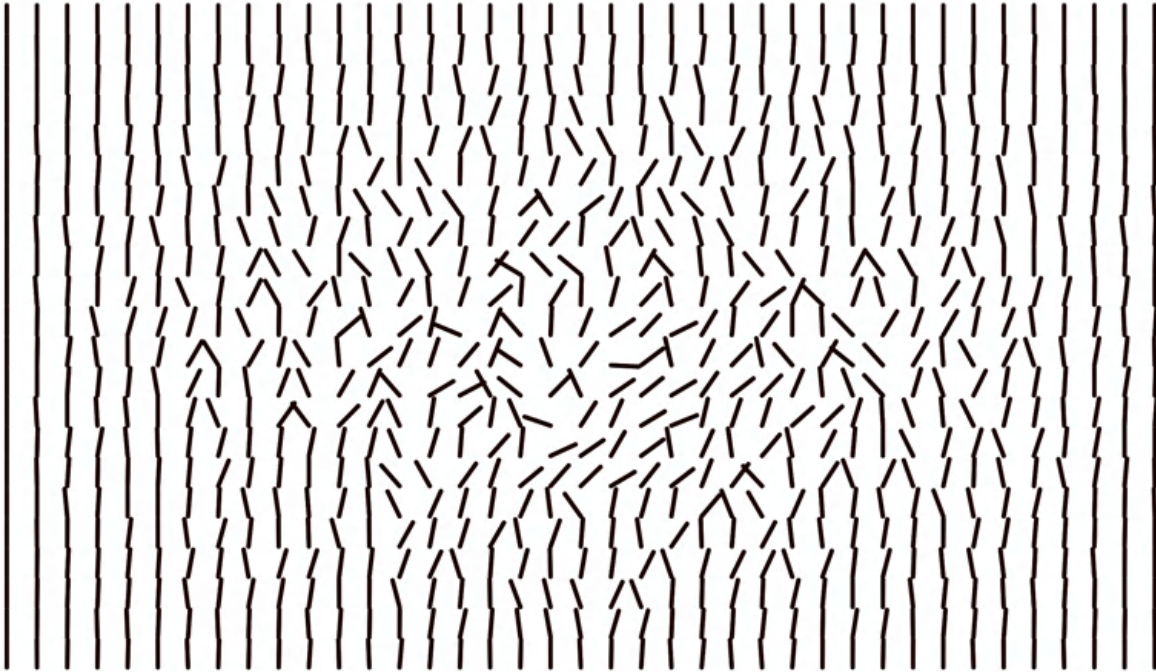
Kleper: Recoding Boxes I, 2012. JW, 2025

Meine eigene Beschäftigung mit *Boxes* hat einerseits zum Recoding der Versionen *Boxes I* und *Boxes II* geführt²¹. Damit lassen sich Bilder nahe an den Originalen von Kolomyjec erzeugen. Auf diese Versionen habe ich verschiedene Formen des Remixing angewandt, gewissermaßen das Remixing des Remixing. Werden etwa - wie bei *Schotter* bereits gezeigt - Polygone mit beliebig vielen Eckpunkten verwendet, lassen sich wieder unterschiedliche Bildeindrücke mit *Boxes* erzeugen.

¹⁹ Von Bill Kolomyjec werden aktuelle Versionen von *Boxes I* and *II* (inzwischen im quadratische Bildformat) angeboten: <https://objkt.com/collections/fxhash/projects/18483>

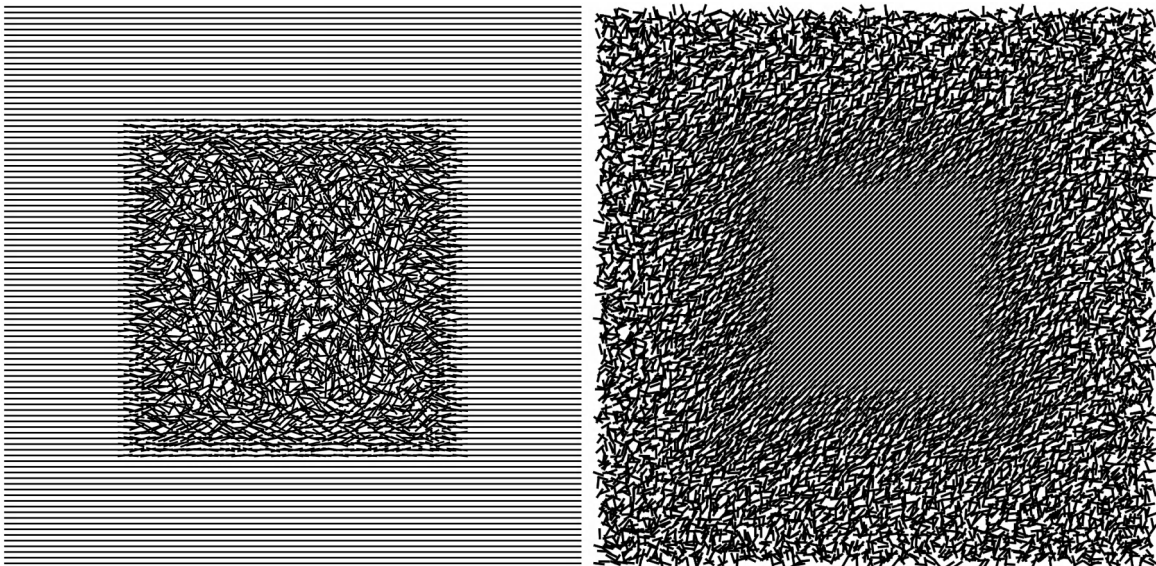
²⁰ Sein Programm läuft direkt im Browser. Der Programmcode ist einsehbar und Kennwerte sowie der Code selbst können geändert werden: https://editor.p5js.org/hendrikleper/sketches/OD_JZNjqw

²¹ Sie können mein Programm selber austesten: *Boxes I* [[Programm3](#)]



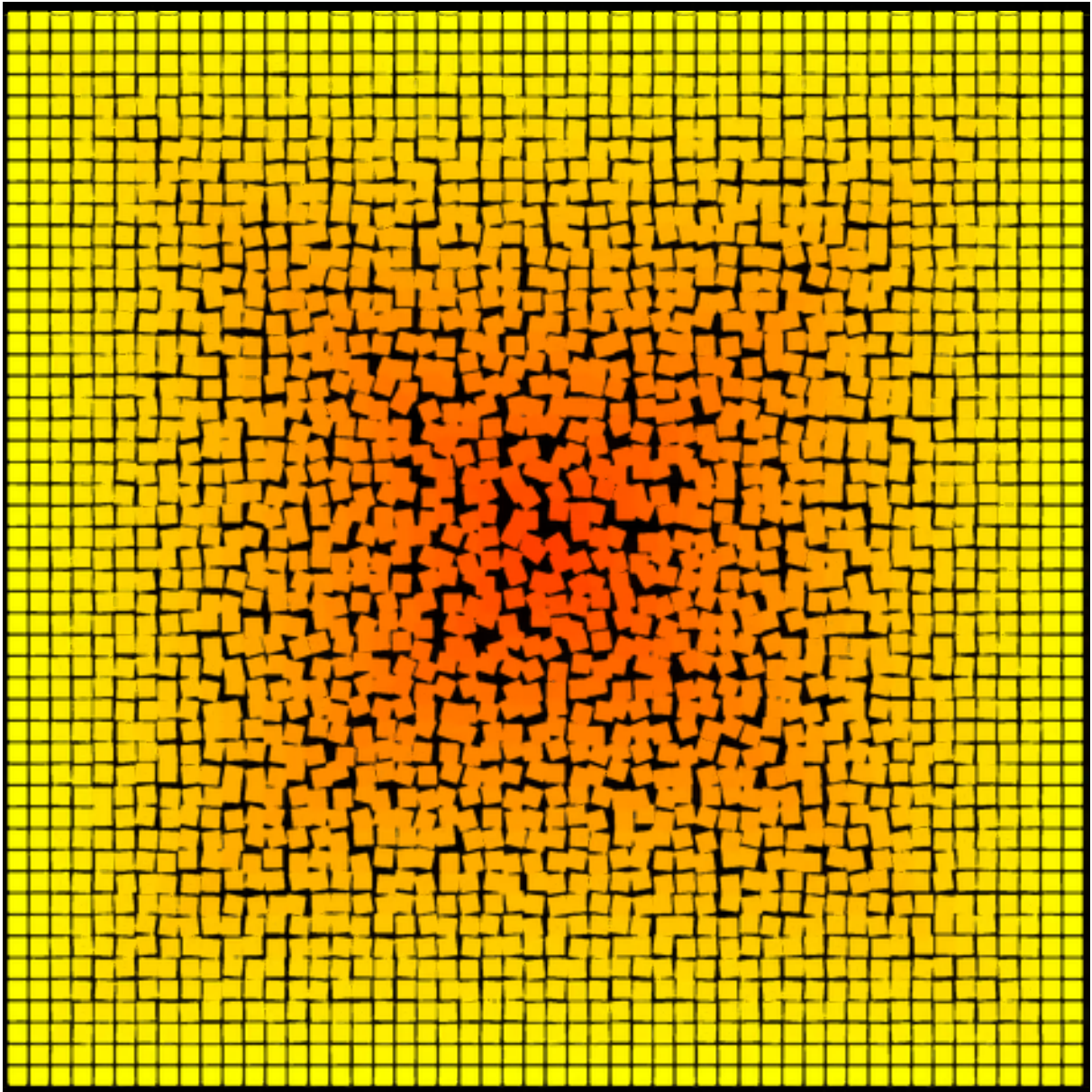
Remixing Boxes I, 2 Eckpunkte. JW 2017

Werden die Zufallsabweichungen von innen gerechnet ab einer bestimmten Position bei den Polygonen unterdrückt, entsteht bei Boxes I ein ebenes „Randplateau“. Im umgekehrten Fall, wenn die Zufallsabweichungen also von außen gerechnet ab einer bestimmten Position unterdrückt werden, entsteht bei Boxes II ein „mittiges Plateau“.



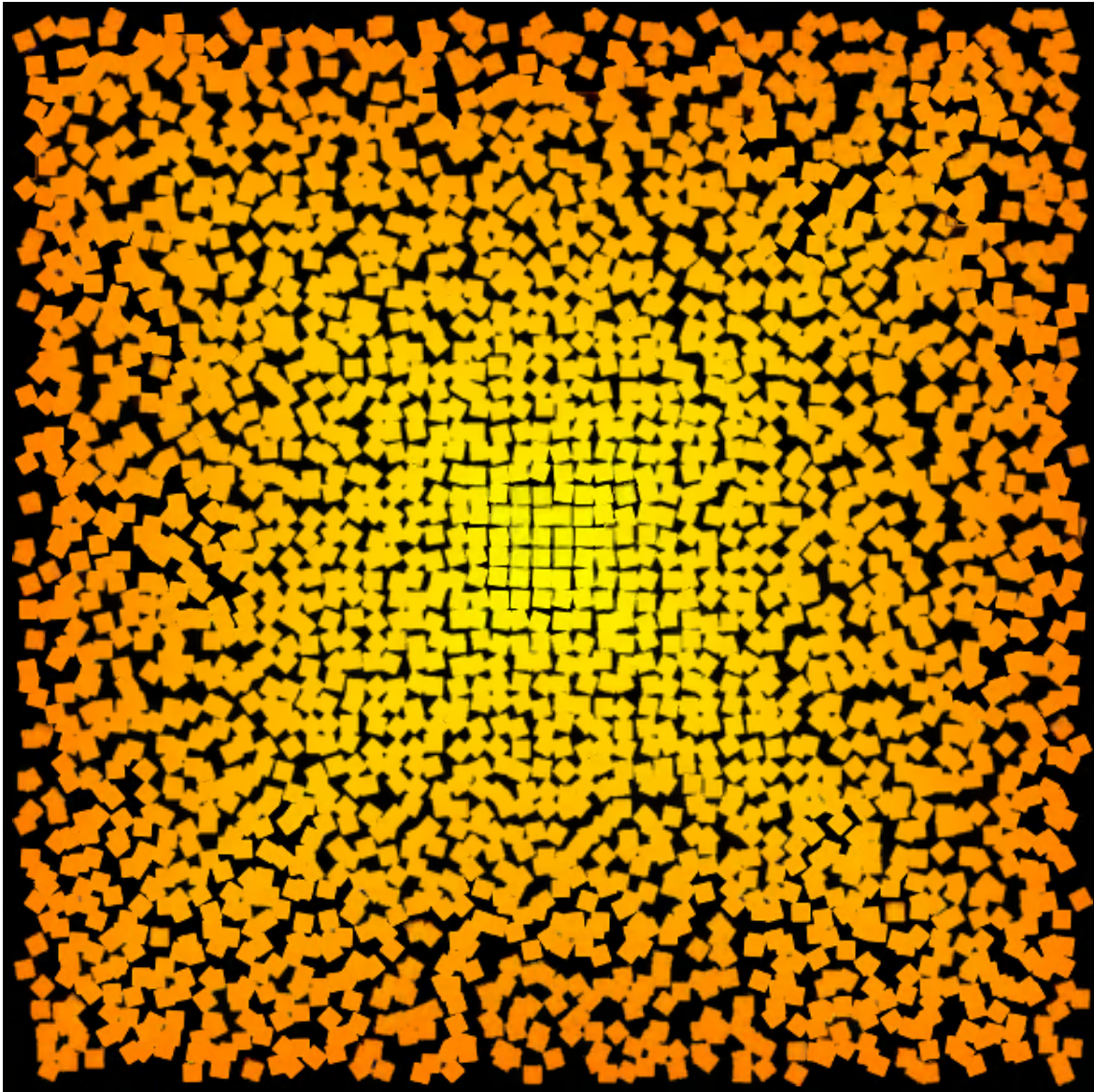
Remixing Boxes I & II: Plateau randständig (links) bzw. mittig (rechts), JW 2017

Werden die Polygone eingefärbt, kann sich ein fast dreidimensionaler Eindruck ergeben. Der kann verstärkt werden durch eine systematische Farbverschiebung, gekoppelt an die Position der Quadrate²².



Remixing Boxes I: Plateau randständig mit Farbverschiebung, JW 2017

²² Sie können mein Programm selber austesten: Boxes I farbig [[Programm 4](#)]



Remixing Boxes II: Plateau mittig mit Farbverschiebung, JW 2017

Schotter: interaktiv & animiert ...

Medienkunst bezeichnet alles künstlerische Arbeiten, das sich der Medien bedient. Da uns beim Recoding und Remixing von *Schotter* besonders die Verwendung digitaler Medien und damit der Einfluss digitaler Technologien auf die Produktion und Rezeption von Kunst interessiert, ist es naheliegend wie Bruce Wands (2006) von *Kunst im digitalen Zeitalter (Art of the Digital Age)* zu sprechen.

Die damit verbundenen neuen Entwicklungen sind - verglichen mit den 60er-Jahren - nicht zu trennen von den drastischen Verbesserungen der inzwischen verfügbaren Werkzeuge und Ausgabemedien. Wie wir an etlichen Beispielen bereits gesehen haben, ist es heute ganz selbstverständlich, auch farbige Darstellungen in hoher Auflösung zu verwenden. Damit sind deutliche Varianten und Erweiterungen der frühen Computerkunst sowie neue Anwendungen realisierbar.

Die Ausgabe der Programmergebnisse ist nicht mehr auf Papier beschränkt. Große Monitore erlauben höchstau aufgelöste, statische und animierte Darstellungen, selbst in Farbe²³.

Möglich geworden ist insbesondere auch die interaktive Änderung von Kenngrößen durch die Betrachtenden sowie die Auswertung von Nutzerdaten mittels Sensoren. Die direkte Reaktion der Programme auf solche von den Betrachtenden erzeugten Daten verwischen zunehmend die Trennung zwischen der technischen Abarbeitung von Algorithmen und der individuellen Gestaltung eines Werkes durch die Künstler selbst und/oder die Betrachtenden.

Auch Frieder Nake - einer der bereits erwähnten Pioniere der Computerkunst - sieht die Zukunft der Algorithmischen Kunst davon geprägt (Nake, 2014):

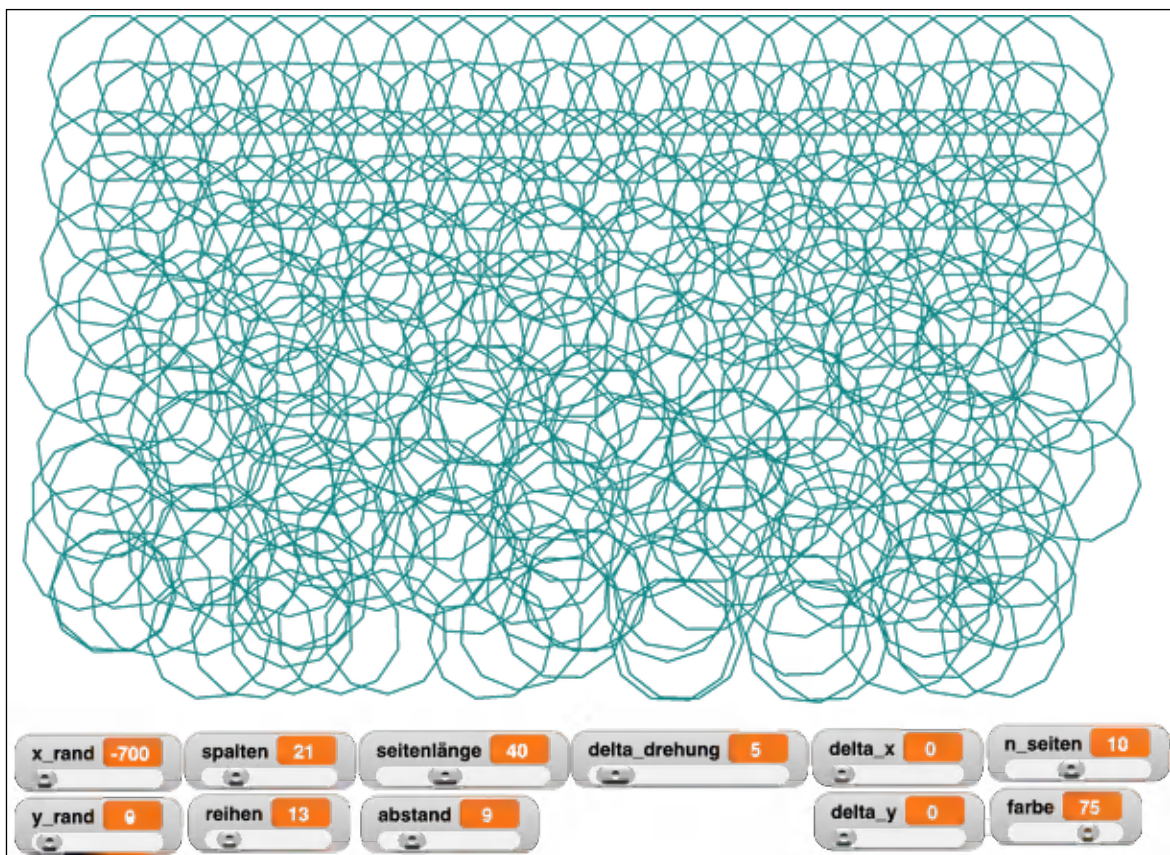
„In jedem Fall muss das Bild in Bewegung sein: Ob als dynamische Installation oder durch Interaktion mit der Umwelt, das ist egal. Das statische Bild ist nicht auf der Höhe des Mediums Algorithmische Kunst. [...] In der weiteren Erforschung des Urproblems – Form und Farbe, die nur zusammen auftreten können – da glaube ich, wird die Zukunft liegen.“

Diese erweiterte Sichtweise kann auch an aktuellen Formen des Remixing von *Schotter* demonstriert werden.

²³ Als Ausgabemedium können sogar ganze Häuserfronten dienen, sei es als Projektionsflächen oder mit programmgesteuerten Farbfenstern, wie es das Projekt *Arbolet frontage* eindrucksvoll zeigen konnte. Eine kurze Videodokumentation finden Sie hier: <https://vimeo.com/205139207>

Bei der Arbeit an einer interaktiven Version von *Schotter* wurde ich auf eine Version des Schweizer Ingenieurs Antoine Beyeler aufmerksam. Seine Nachbildung von *Schotter* erlaubt die Änderung folgender Kenngrößen: Spalten- und Reihenanzahl, Abstand zwischen den Quadraten, Seitenlänge der Quadrate, zufällige Drehwinkel und horizontale/vertikale Verschiebungen sowie die Strichstärke. Alle Änderungen werden sofort umgesetzt, weshalb kontinuierliche Änderungen mit den Schiebereglern quasi animierte Veränderungen erzeugen²⁴.

Meine Version *Schotter total interaktiv* bietet diese und einige zusätzliche Optionen. Mit meinem Projekt möchte ich gleichzeitig verdeutlichen, wie mit dem Prinzip der totalen Parametrisierung ein sehr flexibles Programm entsteht, das den Anwendern vielfältige Experimentiermöglichkeiten bietet²⁵. Über Schieberegler können dieselben Kenngrößen wie bei Beyelers Version verändert werden. Zusätzlich gilt dies für den horizontalen und vertikalen Startpunkt, die Zahl der Eckpunkte (also die Art der Polygone) und die Farbe der Polygone:



²⁴ Sie können Beyerles interaktive Version direkt im Browser austesten: <https://whisk.rs/>

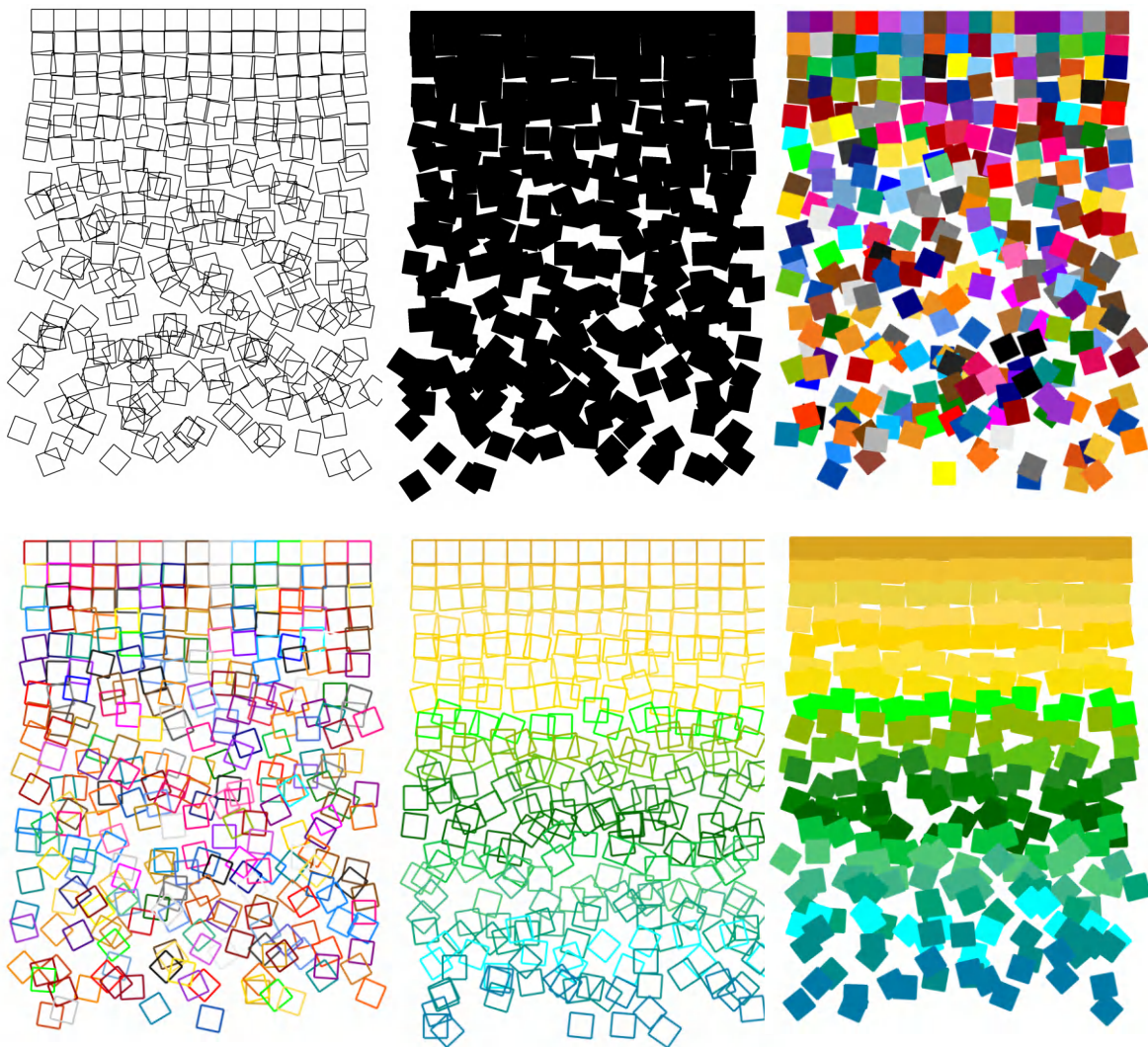
²⁵ Sie können auch mit dieser Version direkt im Browser arbeiten: Schotter total interaktiv [\[Programm 5\]](#)

Alternativ zu den Schiebereglern gibt es etliche Einzelschalter, Mit denen können die Kenngrößen durch Anklicken präzise jeweils um den Wert 1 erhöht oder um -1 erniedrigt werden. Zusätzlich gibt es solche Einzelschalter für weitere Optionen, die die Farben der Polygone betreffen:

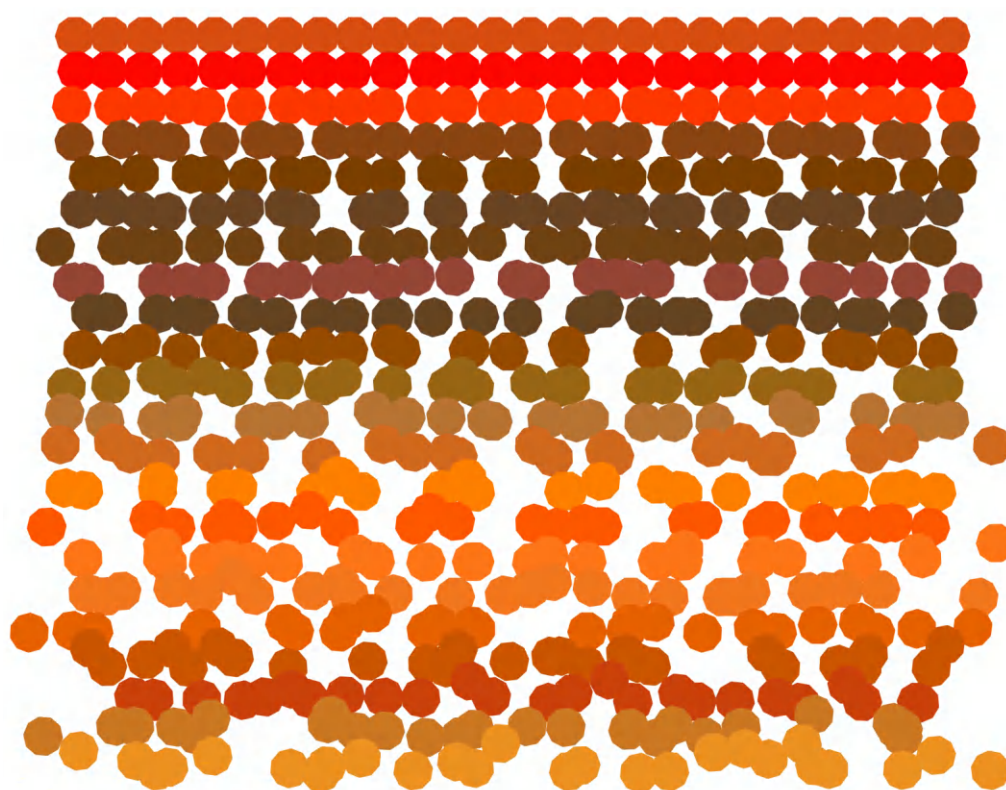
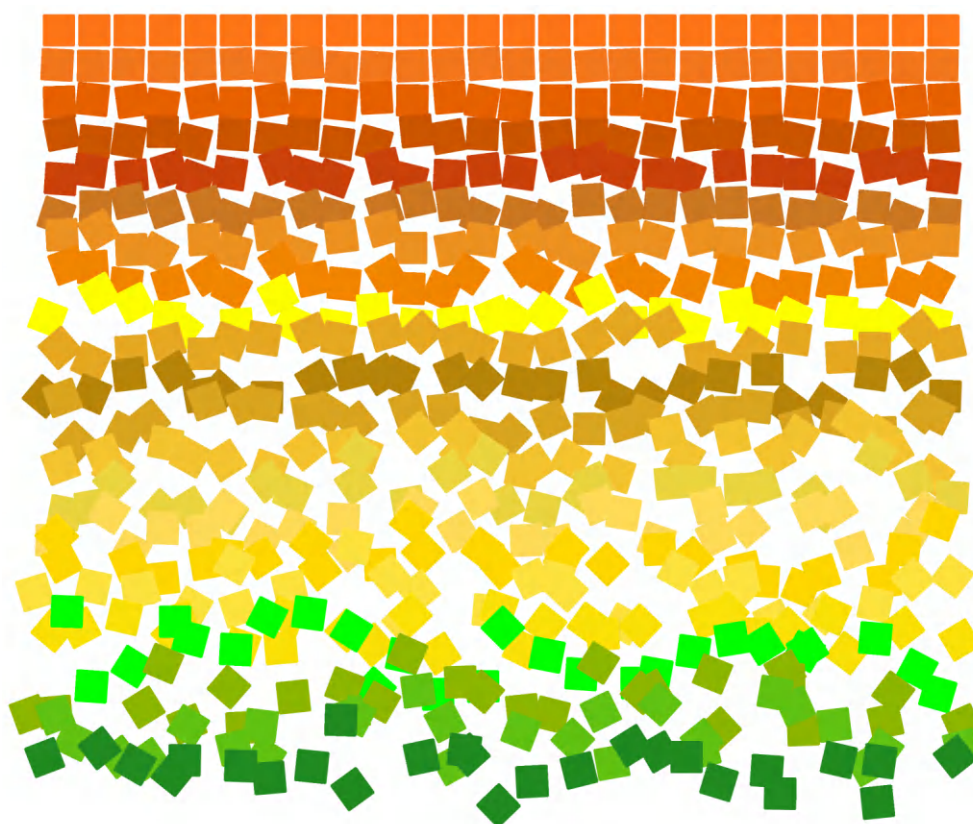
- Ändere Stiftdicke um den Wert 1/-1
- Setze Polygone auf gefüllt/nicht gefüllt
- Setze Einfärbung auf mehrfarbig/einfarbig
- Setze Farbe auf gleiche Farbe pro Reihe
- Ändere Farbwert um den Wert 1/-1
- Setze Stiftfarbe zufällig auf Wert zwischen 0 und 99



Die Bildserie zeigt einige Beispiele zur Nutzung von Farben:



Beispiele von oben links nach rechts unten: Original Schwarz ungefüllt, Schwarz gefüllt, mehrfarbig gefüllt, mehrfarbig nicht gefüllt, Farbe pro Reihe nicht gefüllt, Farbe pro Reihe gefüllt

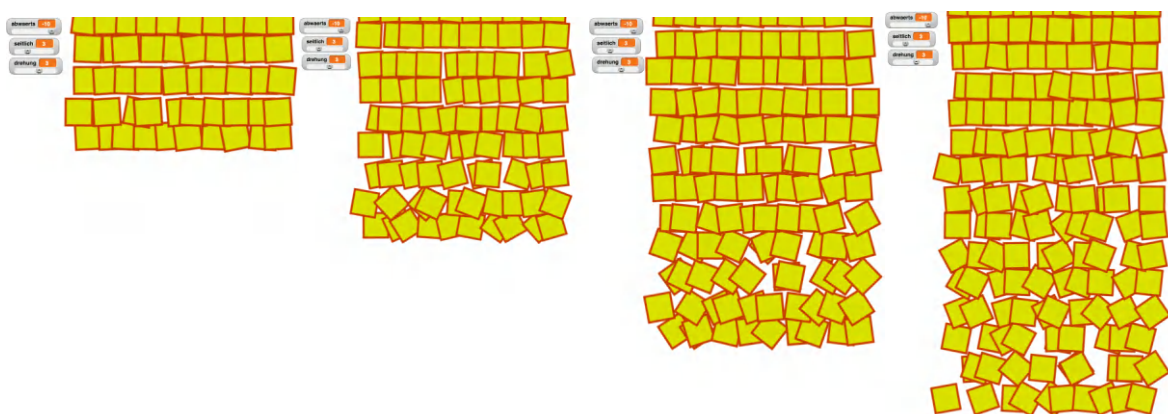


Schotter total interaktiv: 26×22 , Farbe pro Reihe gefüllt, vier Eckpunkte (oben), zehn Eckpunkte (unten), JW 2025

Die gezeigten interaktiven Versionen von *Schotter* weisen schon eine rudimentäre Animation auf, ermöglicht durch die fast verzögerungsfreie Reaktion auf die Eingaben der Betrachtenden. Echte Animation dagegen orientiert sich am Seherlebnis, das wir von Filmen kennen. Filme bestehen aus Einzelbildern, von denen beim professionellen Film typischerweise 24 Bilder pro Sekunde gezeigt werden, um so den Eindruck realistischer Bewegungen zu erzeugen. Computeranimationen sind nun Filme oder entsprechende in Echtzeit ablaufende Bildfolgen, die vollständig am Computer erzeugt und dargestellt werden. Auch in der Computerkunst gehörten von Beginn an solche Computerfilme zum Repertoire der realisierten Formen.

Zu nennen ist Charles Csuri, der die Computerkunst und die Computeranimation von künstlerischer wie auch technischer Seite mit geprägt hat. Bemerkenswert ist, dass für Csuri der Computerfilm bereits damals kein fertiger, vorführbarer Film sein soll, sondern einer, der in Interaktion mit den Benutzern entsteht: *„Auch der Zuschauer kann sich in gewissem Maße als Künstler erleben, wenn er sich ans „Lenkrad“ des Computers setzt. Selbst auf der Grundlage eines vorgegebenen Kontextes sind die Möglichkeiten nämlich fast unendlich groß.“* (Csuri, 1978).

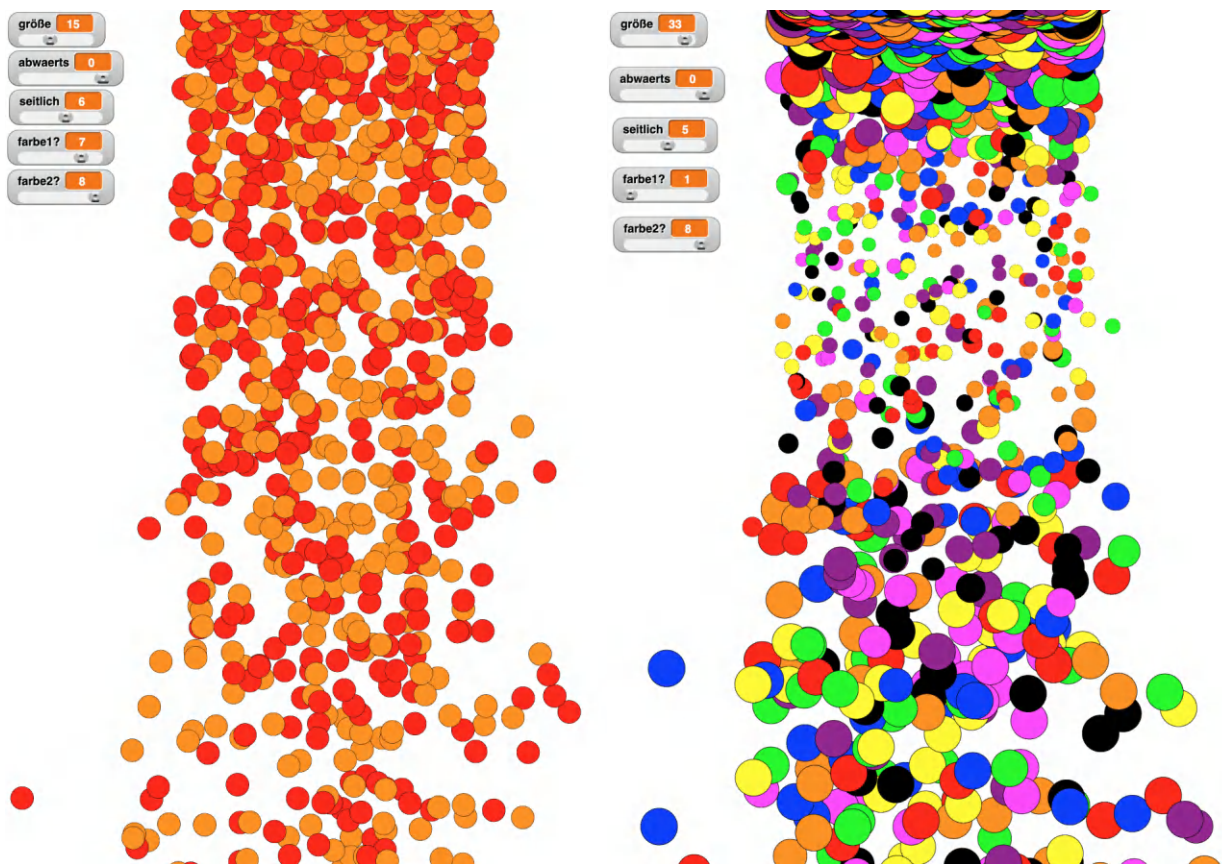
Bei meinem ersten Recoding von *Schotter* hatte ich bereits die ursprüngliche Bildrichtung auf den Kopf gestellt (kaskadierende Quadrate), weil ich den Eindruck von Schotter, der von einem Laster geschüttet wird, erzeugen wollte. Es liegt also nahe, diesen Eindruck durch Animation zu verstärken, orientiert an der originalen Version mit Quadraten²⁶. Die Abbildung unten verdeutlicht dieses Prinzip (naturgemäß nur als statische Abfolge von vier Zwischenstadien). Nebeneinander liegende Steine bewegen sich von oben nach unten. Sie verschwinden am unteren Bildrand, dafür erscheinen von oben immer neue Steine, damit die Animation unterbrechungsfrei abläuft.



²⁶ Sie können diese Version direkt im Browser austesten: Schotter Animation [\[Programm 6\]](#)

Der Eindruck herabfallender Steine wird nochmals verstärkt, wenn farbig ausgefüllte Polygone verwendet werden, wie das vorige Beispiel bereits zeigt. Die Animation ist um das Element der Interaktivität erweitert, denn die Betrachtenden haben die Möglichkeit, den Ablauf der Animation direkt während des Ablaufs zu beeinflussen. Im unten gezeigten Beispiel erlauben dies die Schieberegler **abwärts**, **seitlich** und **drehung**. Das Programm greift ständig auf die aktualisierten Werte zu, reagiert also sofort auf die Eingaben der Betrachter²⁷.

Die Endversion des Programms (gedacht für die Präsentation mit einer interaktiven Station) verwendet vorgefertigte farbige Scheiben²⁸. Über Schieberegler kann deren **größe**, und die Farbe mit **farbe1?** und **farbe2?** fortlaufend verändert werden (für einfarbige Scheiben ist für **farbe1?** und **farbe2?** der gleiche Wert zu wählen).



*Schotter-Scheiben interaktiv: zwei Farben (links), acht Farben, **größe** mehrfach geändert (rechts)*

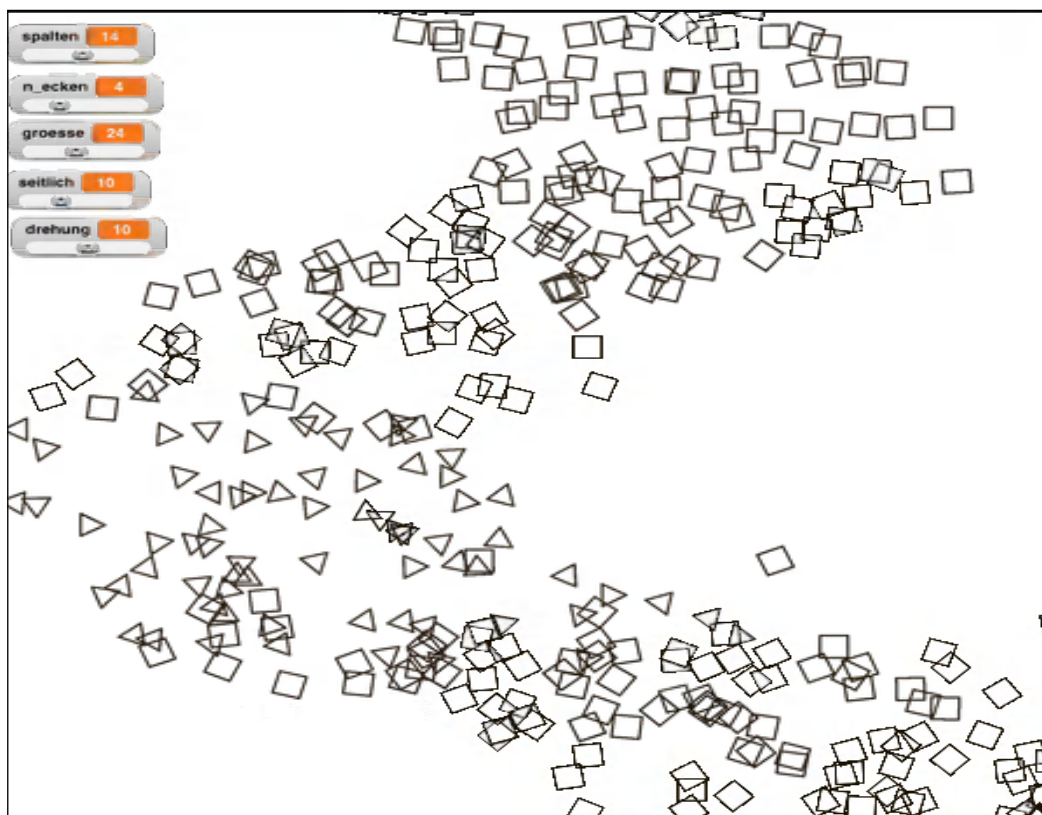
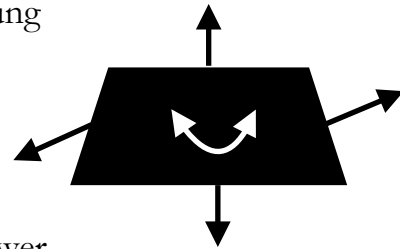
²⁷ Damit die Betrachtenden die Regler mit einer Maus oder mit dem Finger auf einem Touchscreen oder einem Tablet sicher treffen und bewegen können, empfiehlt es sich, die Variablenanzeige zu vergrößern.

²⁸ Sie können diese Version direkt im Browser austesten: Schotter Animation Farbscheiben [[Programm 7](#)]

Auch hier gilt, dass sich die Wirkung der animierten Darstellung in der statischen Abbildung nur erahnen lässt.

Eine noch intuitivere Form der Interaktion ist mit Smartphones oder Tablets möglich, wenn deren Neigungssensoren ausgewertet werden. In einer so erweiterten Form der interaktiven Version von *Schotter* ist folgende Steuerung möglich²⁹:

- Kippen nach rechts oder links ändert die Fallrichtung gegen rechten oder linken Bildrand.
- Kippen nach unten beschleunigt die Fallgeschwindigkeit, Kippen nach oben verzögert sie oder kehrt die Anfangsrichtung um (die Ausrichtungssperre des Tablets sollte dafür am besten aktiviert werden).
- Drehen des Tablets wie ein Lenkrad verändert die Eckenzahl der Polygone.

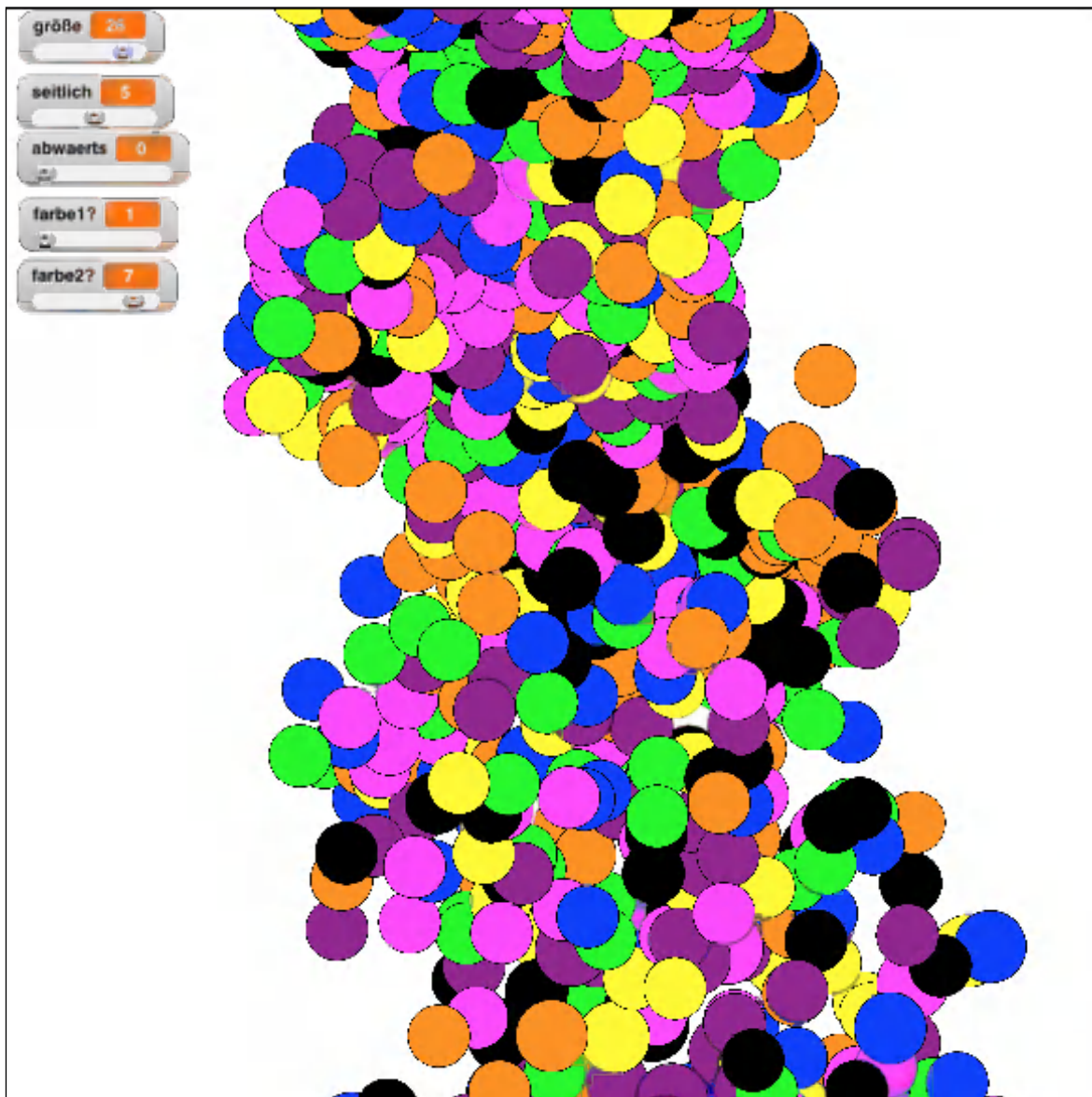


Schotter mit Sensoren: Effekte durch Kippen und Drehen

²⁹ Sie können auch diese Version direkt im Browser austesten (nur mit Tablet oder Smartphone!): Schotter Polygone Sensoren [\[Programm 8\]](#)

Die Änderung der anderen Kennwerte mittels Schieberegler bleibt zusätzlich möglich.

Eine entsprechende Erweiterung der interaktiven *Schotter-Scheiben* Version arbeitet nach demselben Prinzip. Allerdings wird hier durch Drehen des Geräts die Kenngröße **seitlich** verkleinert und vergrößert³⁰.



Schotter-Scheiben mit Sensoren: Effekte durch Kippen und Drehen

³⁰ Sie können auch diese Version direkt im Browser austesten (nur mit Tablet oder Smartphone!): Schotter Farbscheiben Sensoren [\[Programm 9\]](#)

Bilder denken, nicht malen ...

Mit der kleinen Retrospektive auf 60 Jahre Computerkunst am Beispiel *Schotter* habe ich hoffentlich zeigen können, wie wichtig es bei allen Formen der Algorithmischen und Generativen Kunst ist, dass wir *Bilder denken* können, *nicht malen* müssen. Dafür sollten wir verstehen, welche Rolle die Maschinen im Entstehungsprozess der Bilder spielen und ob und wie die Rolle der Kunstschaffenden sich dadurch ändert.

Am Beispiel *Schotter* haben wir sehen können, wie die Analyse einer auf den ersten Blick ziemlich einfachen Bildstruktur sich als äußerst anregend herausstellen kann. Das Recoding & Remixing der frühen Computerkunst ließe sich vergleichbar an zahlreichen Werke dieser Phase fortsetzen. Der niedrigschwellige Zugang zu dafür nutzbaren Werkzeugen, wie sie heute verfügbar sind, lädt Sie vielleicht ein, sich daran zu beteiligen.

Bei der Ausstellung meiner eigenen Werke zur Computerkunst wurde ich unter anderem gefragt, was ich denn bei den Exponaten selber gemacht hätte. Für mich ist meine Rolle dabei klar. Ich habe dafür tatsächlich keinen Zeichenstift oder Pinsel in die Hand genommen. Das ändert aber nichts daran, dass nach wie vor ich selber und nicht die Maschine für die Ausprägungen der durch meine Algorithmen erzeugten Bilder - als Repräsentanten der jeweiligen Klasse aller denkbar möglichen Bilder - verantwortlich bin. Darin unterscheidet sich dieses Arbeiten grundlegend von der *Generativen KI*, die derzeit innerhalb und außerhalb des Kunstbetriebs hohe Aufmerksamkeitswellen schlägt. Aber das ist ein anderes Thema ...

Joachim Wedekind

Literatur

- Csuri, C. (1978). Real Time Picture Animation. In: IBM Deutschland GmbH (Hrsg.): Computerkunst. S. 62 f. Stuttgart: IBM Deutschland GmbH.
- Franke, H. W. (1971). Computergraphik Computerkunst. München: Bruckmann.
- Groos, U. & Froitzheim, E.-M. (Hrsg.) (2016). [un]erwartet. Die Kunst des Zufalls. Köln: Wienand Verlag.
- hex6c (2020). Generative Art Recoded. Medium 13. April. <https://hex6c.medium.com/generative-art-recoded-675c3eabb2f2>
- Hunter, Z. (2024-1). Schotter - Georg Nees - Part 1: Introduction. <https://zellyn.com/2024/06/schotter-1/>
- Hunter, Z. (2024-2). Schotter - Georg Nees - Part 2: Investigation. <https://zellyn.com/2024/06/schotter-2/>
- Kolomyjec, W.J. (1975). The Appeal of Computer Graphics. In: R. Leavitt (Ed.): Artist and Computer. pp. 45–51, New York: Harmony Books. Download unter <http://www.atararchives.org/artist/sec15.php>
- Nake, F. (1995). Künstliche Kunst im Strom der Zeit. Vortrag Berliner Sommer-Uni '95. Download unter <http://radicalart.info/AlgorithmicArt/Nake95.html>
- Nake, F. (2014). „No Message Whatsoever“ (Interview). HPImgzn Ausgabe 15, S. 11-13. Potsdam: HPI. Download unter: https://hpi.de/fileadmin/user_upload/hpi/dokumente/hpi_mgzn/HPImgzn_Ausgabe15.pdf
- Nake, F. (2016). The Disappearing Masterpiece Digital Image & Algorithmic Revolution. In: M. Verdicchio et al. (Ed.): xCoAx 2016 Proceedings, pp. 12-27. Bergamo. <http://2016.xcoax.org/xcoax2016.pdf>
- Nake, F. (2021). $I(\chi, \chi') = g(\chi, \chi') [e(\chi, \chi') + \int p(\chi, \chi', \chi'') I(\chi', \chi'') dx'']$. In Diestelmeyer, Ehrmanntraut & Müller (Hg.): Algorithmen & Zeichen. Beiträge von Frieder Nake zur Gegenwart des Computers. Berlin: Kulturverlag Kadmos.
- Nake, F. & Grabowski, S. (2017). Think the Image, Don't Make It! On Algorithmic Thinking, Art Education, and Re-Coding. Journal of Science and Technology of the Arts, Volume 9, No. 3. <https://revistas.ucp.pt/index.php/jsta/article/view/7292/7292>
- Nees, G. (1969). Generative Computergraphik. Ein Nachdruck mit einleitenden Texten von Herrmann und Nees (2006) ist als eText erhältlich: <http://www.diaphanes.net/titel/generative-computergraphik-1948>
- Nees, G. (1995). Formel, Farbe, Form. Computerästhetik für Medien und Design. Heidelberg: Springer.
- Nees, G. (2005). Künstliche Kunst. Die Anfänge. Ausstellungsbroschüre (unpaginiert). Bremen: Kunsthalle.
- Plaxco, J. (2006). Georg Nees, Processing, and a Schotter Tutorial. https://www.artsnova.com/Nees_Schotter_Tutorial.html
- Rauch, H. & Wedekind, J. (1989). Schildkrötengrafik. Einfaches Programmieren von Grafiken. Lehren und Lernen mit dem Computer 13. Tübingen: DIFF.
- Schneeberger, R. (2013). Programmierte Fälschungen Digitaler Kunst als Museumsprojekt? Telepolis (Online-Journal). Haar: Heise Medien.

- Sol Le Witt (1967). Paragraphs on conceptual art. *Artforum* 5, 10 (June 1967), 79–83.
- Stalder, F. (2009). Neun Thesen zur Remix-Kultur. https://irights.info/wp-content/uploads/fileadmin/texte/material/Stalder_Remixing.pdf
- Stein, H. (1984). LOGO - Grafik, Sprache, Mathematik. Haar Bei München: Markt & Technik Verlag.
- Toft, T. (2014). An intro to Pen Plotters. Medium <https://medium.com/quarterstudio/an-intro-to-pen-plotters-29b6bd4327ba>
- Wands, B. (2006). *Art of the Digital Age*. London: Thames & Hudson.
- Wedekind, J. (2018). Codierte Kunst. Kunst Programmieren mit Snap! Tübingen: Selbstverlag. <http://digitalart.joachim-wedekind.de/my-books/>
- Zeilinger, M. (2018). Plotting Critical Research-Practice in Digital Art. In: Golding (Ed.): *Parsing Digital - Conversations in digital art by practitioners and curators*, S. 22-37. London: Austrian Cultural Forum. https://www.academia.edu/39607548/Plotting_Critical_Research_Practice_in_Digital_Art

Bildernachweise

- S. 3: Schotter: Nees (1969), Bild 38, S. 242
- S. 5: Georg Nees: Foto Alexander Kempkens, 1986; https://commons.wikimedia.org/wiki/File:Georg_Nees-1- BID_1986.jpg
- S. 6: ZUSE Graphomat Z 64, Zeichentisch & Steuerelektronikschrank; Leihgabe des Deutschen Technikmuseums Berlin an DAM Galerie Berlin, Ausstellung Aesthetica, 2015; Fotos: Inge Wedekind
- S. 12: Schotter: Victoria and Albert Museum (Kensington, GB); <https://collections.vam.ac.uk/item/O221321/schotter-print-nees-georg/>
- S. 12: Schotter: The Anne and Michael Spalter Digital Art Collection; <https://spalterdigital.com/artworks/schotter-gravel-stones/>
- S. 37: Schotter Ilyá Belsky, <https://observablehq.com/@oluckyman/georg-nees-schotter?collection=@oluckyman/generative-art>
- S. 38: Jim Plaxco: Cubic Disarray: Turbulence, <https://www.artsnova.com/art-Cubic-Disarray-Turbulence.html>
- S. 38: Jim Plaxco: Cubic Disarray: Bisection, <https://www.artsnova.com/art-Cubic-Disarray-Bisection.html>
- S. 40: hex6c: Schotter - Take I, <https://superrare.com/artwork/eth/0xb932a70a57673d89f4acffbe830e8ed7f75fb9e0/9453>
- S. 41: hex6c: Schotter - Take II, <https://superrare.com/artwork/eth/0xb932a70a57673d89f4acffbe830e8ed7f75fb9e0/9454>
- S. 41: hex6c: Schotter - Take III, <https://superrare.com/artwork/eth/0xb932a70a57673d89f4acffbe830e8ed7f75fb9e0/9487>
- S. 41: hex6c: Schotter - Take IV, <https://superrare.com/artwork/eth/0xb932a70a57673d89f4acffbe830e8ed7f75fb9e0/9488>
- S. 42: Boxes I & II: <https://www.drbillkolomyjec.com/hard-copy/plotter-prints>

Meine *Schotter*-Programme

- S. 13 [Programm 1] Hommage to the Square: <https://snap.berkeley.edu/snap/snap.html#present:Username=jowede&ProjectName=Hommage%20To%20The%20Square&editMode&noRun>
- S. 17 [Programm 2] Schotter Nees Original: <https://snap.berkeley.edu/snap/snap.html#present:Username=jowede&ProjectName=Schotter%20Nees%20Original%20recoded>
- S. 43 [Programm 3] Boxes I: <https://snap.berkeley.edu/snap/snap.html#present:Username=jowede&ProjectName=Hommage%20Kolomyjec%20Boxes%201>
- S. 45 [Programm 4] Boxes I farbig: <https://snap.berkeley.edu/snap/snap.html#present:Username=jowede&ProjectName=Hommage%20Kolomyjec%20Boxes%201%20mit%20Rande%20bene%20farbig>
- S. 48 [Programm 5] Schotter total interaktiv: https://snap.berkeley.edu/snap/snap.html#present:Username=jowede&ProjectName=schotter_total_interaktiv
- S. 51 [Programm 6] Schotter Animation: <https://snap.berkeley.edu/snap/snap.html#present:Username=jowede&ProjectName=Schotter%20Animation>
- S. 52 [Programm 7] Schotter Animation Farbscheiben: <https://snap.berkeley.edu/snap/snap.html#present:Username=jowede&ProjectName=Schotter%20Animation%20Farbscheiben>
- S. 53 [Programm 8] Schotter Polygone Sensoren: <https://snap.berkeley.edu/snap/snap.html#present:Username=jowede&ProjectName=Schotter%20Polygone%20Sensoren>
- S. 54 [Programm 9] Schotter Farbscheiben Sensoren: <https://snap.berkeley.edu/snap/snap.html#present:Username=jowede&ProjectName=Schotter%20Farbscheiben%20Sensoren>

Zur Person

Joachim Wedekind war Unterrichtstechnologe und Mediendidaktiker. Als Spät- und Quereinsteiger lotet er heute unter engem Bezug zur frühen Computerkunst die Rolle der Malmaschinen aus. Dies war Schwerpunkt mehrerer Ausstellungen, die auch im Netz dokumentiert sind:

<http://digitalart.joachim-wedekind.de/exhibitions/>

Seine Erfahrungen beim Programmieren Generativer Kunst hat er im Buch *Co-dierte Kunst Kunst - Programmieren mit Snap!* zusammengefasst. Herausgekommen ist dabei eine Mischung aus Bildband und Programmierleitfaden:

<http://digitalart.joachim-wedekind.de/ueber-das-buch/>