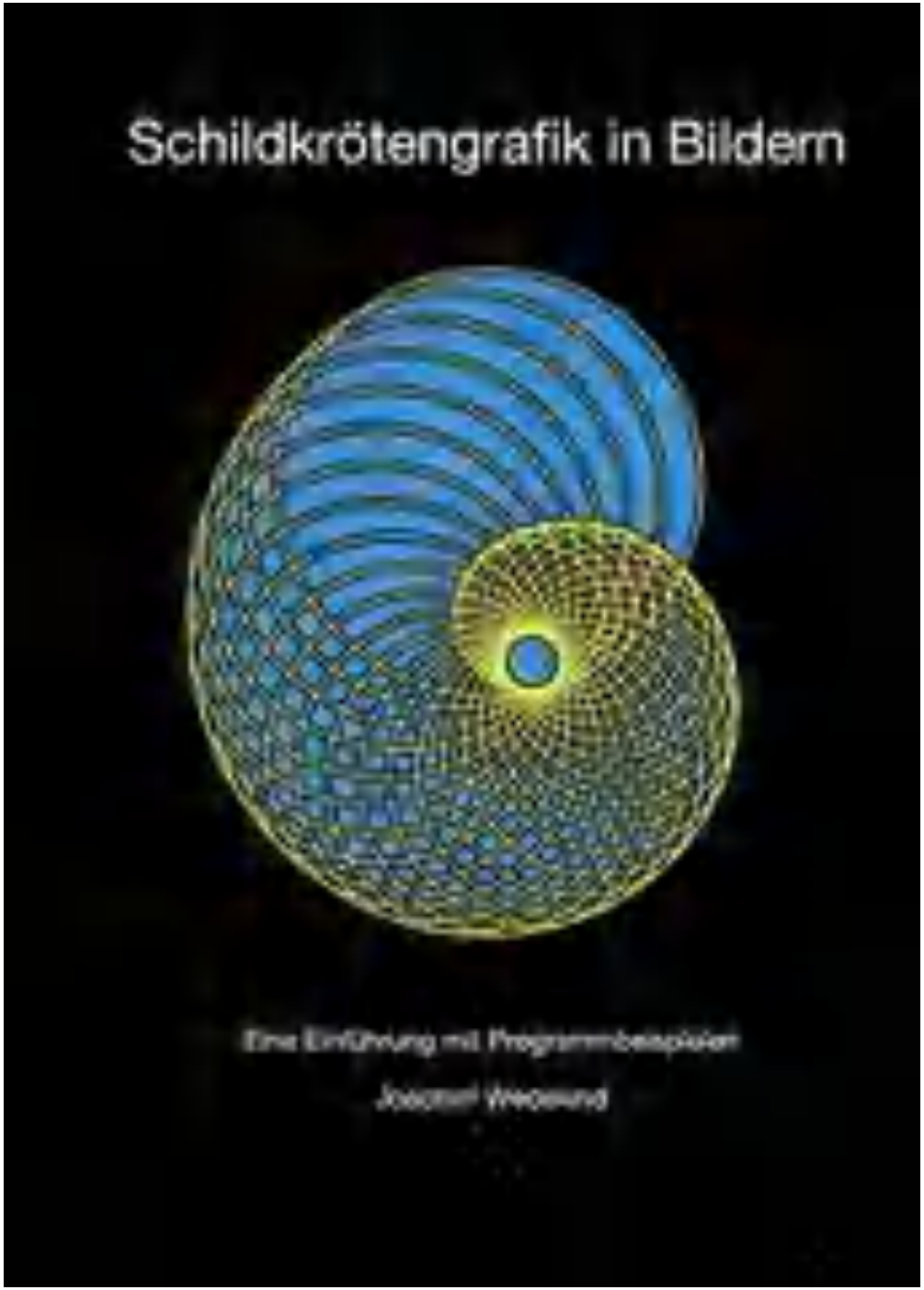

Schildkrötengrafik in Bildern
Eine Einführung mit Programmbeispielen

Schildkrötengrafik in Bildern

Eine Einführung mit Programmbeispielen

Joachim Wedekind

Joachim Wedekind:
Schildkrötengrafik in Bildern. Eine Einführung mit Programmbeispielen

Autor: Dr. Joachim Wedekind
Eschenweg 26
72076 Tübingen

Webseite zum Buch:
http://programmieren.joachim-wedekind.de/logo-klassiker/

2020 Tübingen

Grafische Gestaltung und Satz: Joachim Wedekind

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

Vorwort

Kurz nach meinem beruflichen Einstieg in die Unterrichtstechnologie und Mediendidaktik las ich 1975 zum ersten Mal in einem Übersichtsband *„Computer und Unterricht“* (Eyferth et al., 1974) über das Logo-Projekt von Papert am MIT. Von den Autoren *„nach seinen curicularen und didaktischen Zielsetzungen als einer der bemerkenswertesten Versuche sinnvoller Computerverwendung im Unterricht“* angesehen, versuchten sie dies anhand typischer Programmbeispiele zu belegen. Diese zeigten einfache geometrische Strichfiguren, die mit der „Turtle“ (Schildkröte), einem steuerbaren Roboter, gezeichnet wurden.

Diese Bilder waren und sind typisch für den Einsatz von Logo im schulischen Kontext. Sie finden sich in frühen Publikationen zu Logo (den Handbüchern zu verschiedenen Logo-Versionen und vielen einführenden Büchern). Die Bilder und die Möglichkeiten für ihre Erzeugung haben mich sofort angesprochen. Für mich besitzen sie eine ganz eigene Ästhetik mit einem hohen Erkennungswert in den entsprechenden Publikationen.

Dieses Buch stellt deshalb typische mit der Schildkrötengrafik erzeugte Bilder in den Mittelpunkt. Bei manchen Beispielen wird die Nähe zu Werken der frühen *Computerkunst* (die ich als Hommagen an die Künstler aufgenommen habe), aber auch zu bekannten optischen Täuschungen (die ich als *Opticals* bezeichne), sehr deutlich.

Es werden jeweils nur grundlegende Programmierkonzepte knapp skizziert, die für die Erzeugung der jeweiligen Bilder verwendet werden. Das Buch ist damit also keine Einführung in das Programmieren mit Logo, sondern eher ein Bilderbuch, das vielleicht gerade dadurch anregen kann, selber diese und vergleichbare Bilder selber zu programmieren.

Tübingen, April 2020 Joachim Wedekind

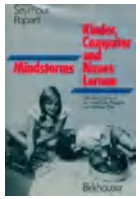
Werbung in eigener Sache: Zur *Computerkunst* und zu den *Opticals* gibt es zwei Bücher und zugehörige Webseiten von mir, die dem Programmieren den angemessenen Stellenwert einräumen:

J. Wedekind (2018). Codierte Kunst. http://digitalart.joachim-wedekind.de/buchbestellung/

J. Wedekind (2019): Optische Täuschungen animieren für Dummies Junior. http://opticals.joachim-wedekind.de/das-buch/

Inhalt

Einführung: Logo Klassiker



Nach Jahren habe ich das Buch Mindstorms von Seymour Papert wieder entdeckt (Papert, 1980, 1982)[1]. Er entfaltet darin seine Sicht des Computers als geistiges Werkzeug und wie Kinder sich dieses Werkzeug mit Hilfe von Logo erschließen können.

Als zentrale Komponente von Logo wurde ab 1970 die Schildkrötengrafik (Turtle graphics) eingeführt, durch die mit wenigen Grundbefehlen einfache, aber auch komplexe ansprechende Grafiken erzeugt werden können[2].

„Die Schildkrötengeometrie ist ein anderer Stil der Geometrie. Euklids Stil ist ein logischer, Descartes' Stil ist ein algebraischer. Schildkrötengeometrie ist ein algorithmisierter Stil der Geometrie." (Papert, 1982, S. 84)

Anhand von Code-Beispielen kann die Entstehung dieser Bilder leicht nachvollzogen werden[3]. Wer Freude an abstrakt-geometrischen Grafiken hat, wird vielfältige Anregungen finden, wie mit einfachen programmiertechnischen Mitteln ansprechende Bilder erzeugt werden können.

[1] Die Lektüre lohnt sich auch und gerade heute noch, wenn überlegt wird, ab wann und in welcher Form Informatikunterricht in der Schule eingeführt wird. Die englische Originalfassung des Buchs kann kostenlos herunter geladen werden.

[2] Für die grafische Ausgaben wurde ein schildkrötenähnlicher Roboter entwickelt: die *Yellow Turtle* (Feurzeig, 2010). Dieses Gefährt konnte sich vorwärts bewegen und drehen sowie einen Schreibstift anheben und senken. Auf einem darunter liegenden Papier können so Linien gezogen werden. Später wurde der Roboter durch ein kleines Symbol mit Richtungsanzeiger auf dem Bildschirm abgelöst.

[3] Die Beispiele sind mit der visuellen Programmierumgebung Snap! umgesetzt, quasi einer „Enkelin" von Logo.

Der Ansatz, der der Schildkrötengrafik zugrunde liegt, wird für mich nach wie vor am besten von Seymour Papert selbst in *Mindstorms* beschrieben. Seine Grundprinzipien finden sich kompakt dargestellt und illustriert in Kapitel 3: *Schildkrötengeometrie: Eine Mathematik, die fürs Lernen gemacht ist*. Sie sollen hier mit interaktiven Beispielen illustriert werden.

HInweis: Zum Verständnis der Beispiele ist es empfehlenswert, sich den vollständigen Code der Beispiele anzusehen. Die Links dazu finden Sie auf der Website Logo Klassiker. Dort führt ein Klick auf den **Programmnamen** jeweils direkt zu den Programmen in der Snap!-Programmierumgebung.

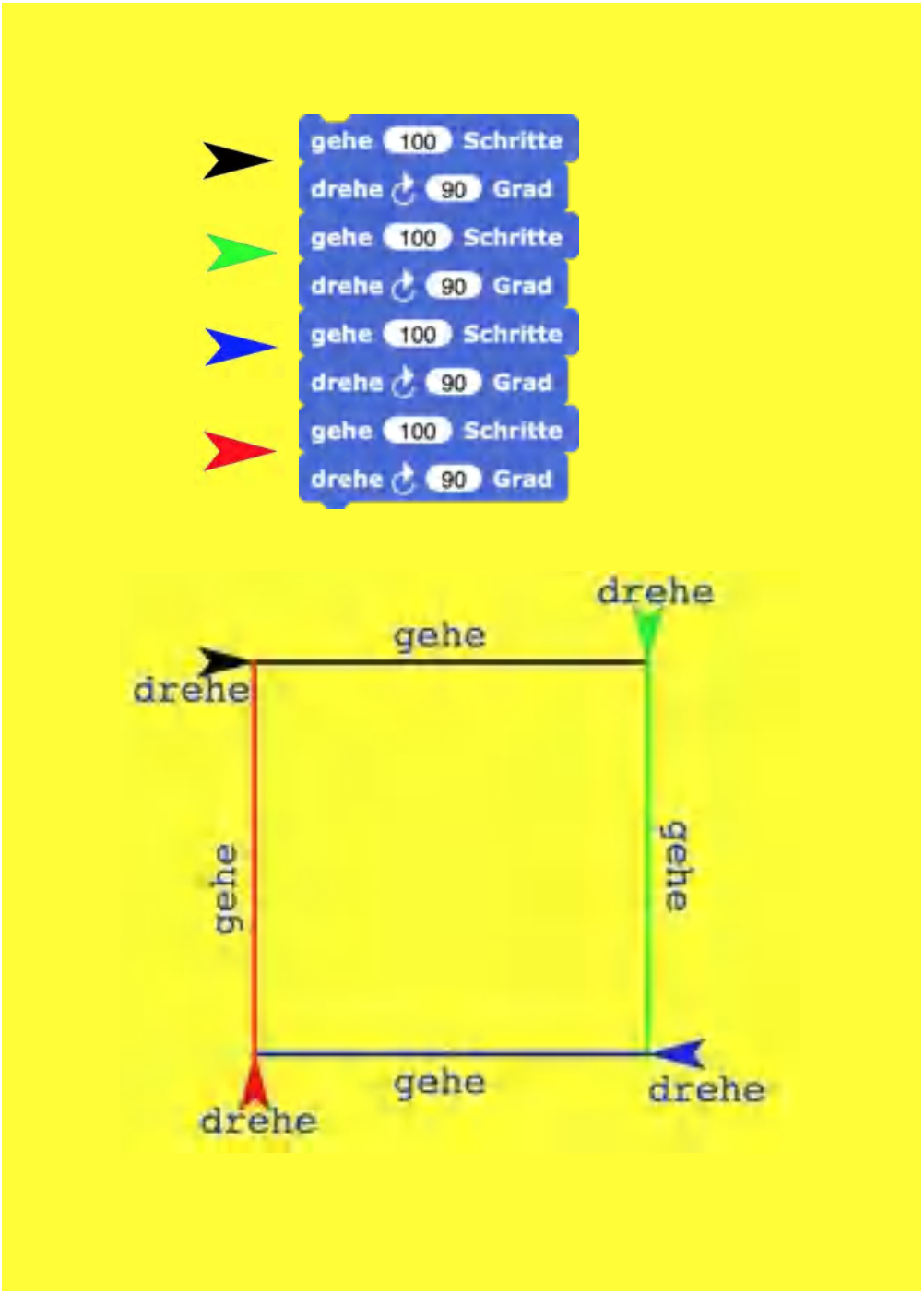
Wird **Snap!** neu gestartet, öffnet es sich mit einem vorgegebenen **Anfangszustand**: Die **Bühne** hat eine bestimmte **Breite** und **Höhe**, ist **leer** und hat die Farbe **Weiß**. Die **Schildkröte** sitzt in der **Mitte** der Bühne in der Form eines **Pfeils**, zeigt nach **rechts** und hat die Farbe **Schwarz**. Alle diese Eigenschaften lassen sich nachträglich leicht ändern.

Die Schildkröte[4] kann Befehle in der sogenannten *Schildkrötensprache* verstehen[5]. Damit wird ihre **Bewegung** kontrolliert. Mit dem Befehl **gehe** bewegt sie sich in einer geraden Linie in ihrer Blickrichtung, d.h. sie verändert ihre Position, aber nicht ihre Blickrichtung. Mit dem Befehl **drehe** verändert sie ihre Blickrichtung, aber nicht ihre Position.

Im gegenüberliegenden Beispiel werden die Bewegungen der Schildkröte durch solche Befehle beschrieben und damit ein Quadrat gezeichnet:

[4] In deutschsprachigen Logo-Einführungen (z.B. Abelson, 1983; Hoppe & Löthe, 1984 oder Ziegenbalg, 1985) und in den deutschen Versionen der Programmiersprache Logo (aktuell in MSWLogo) wurde für *Schildkröte* der Ausdruck *Igel* (wegen der Kürze des Wortes) eingeführt.

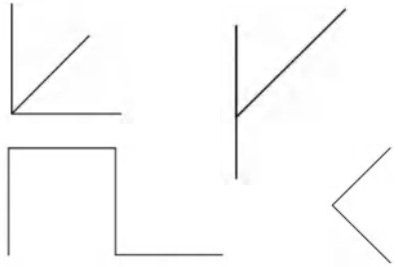
[5] Ich verwende hier die deutschen Befehle. In Snap! kann aber bei Bedarf leicht auf andere Sprachen umgeschaltet werden.



Liniengrafiken

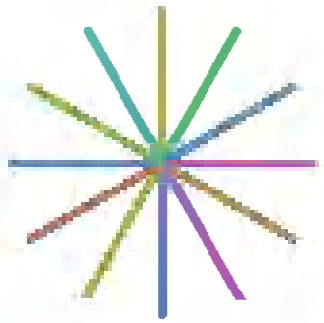
Damt die Schildkröte zeichnet, muss ihr Zeichenstift mit **Stift runter** abgesenkt werden. Erst dann hinterlässt sie eine Spur. Mit **Stift hoch** kann der Zeichenstift wieder angehoben werden.

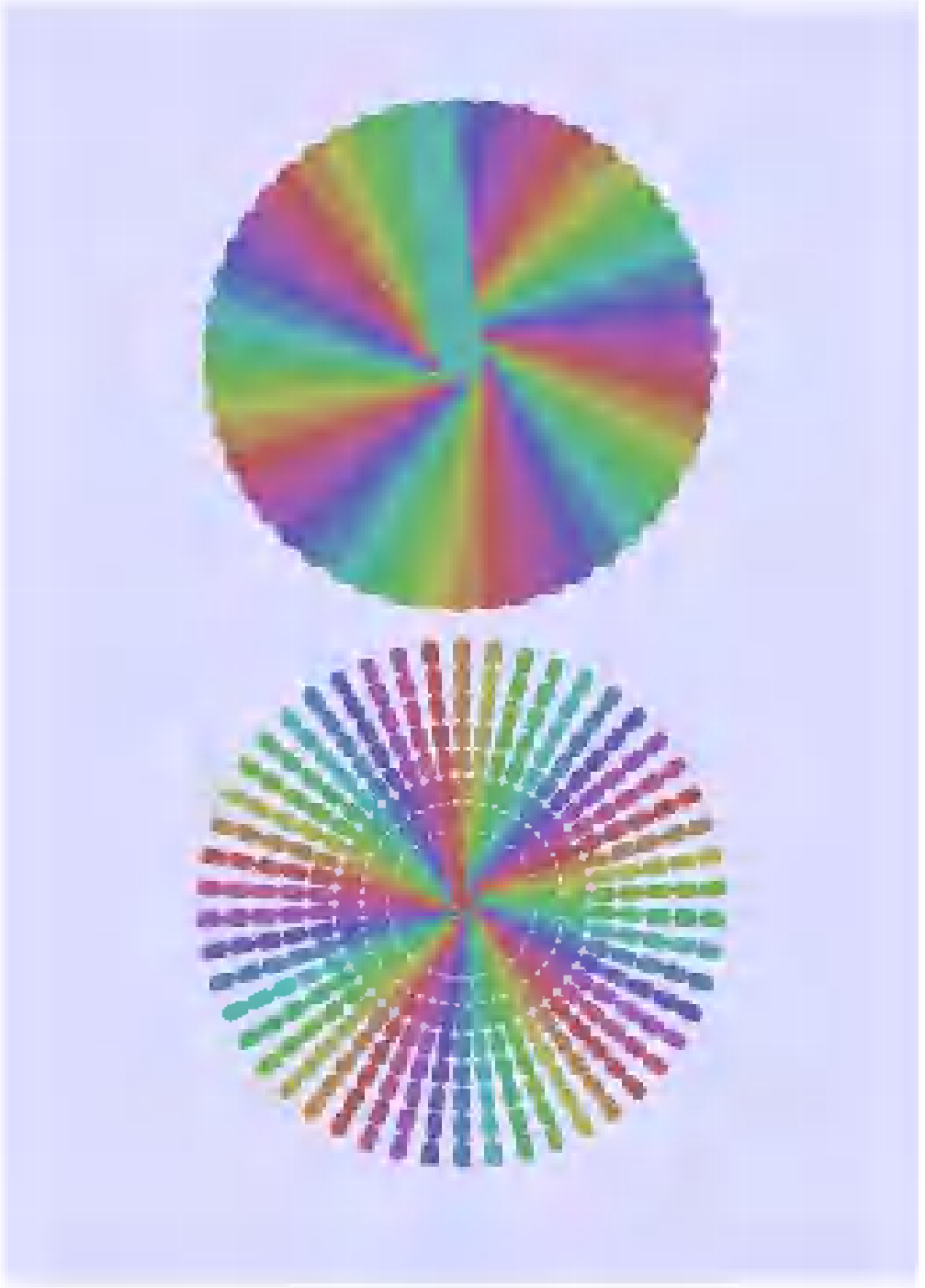
Allein mit den Bewegungen **gehe** und **drehe** sowie **Stift runter** und **Stift hoch** sind bereits viele unterschiedliche Grundmuster möglich.



Wiederholungen: Der Computer ist besonders gut darin, häufig wiederkehrende, stumpfsinnige, aber notwendige Aktionen geduldig und schnell auszuführen. Dafür gibt es u.a. in Snap! den **wiederhole**-Block, dessen darin eingeschlossenen Befehle so oft ausgeführt werden wie angegeben.

Werden die Grundmuster kombiniert und in den **wiederhole**-Schleifen vervielfacht, lassen sich damit bereits viele ansprechende Liniengrafiken erzeugen.

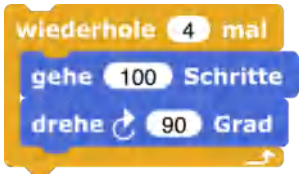




Quadrate (I)

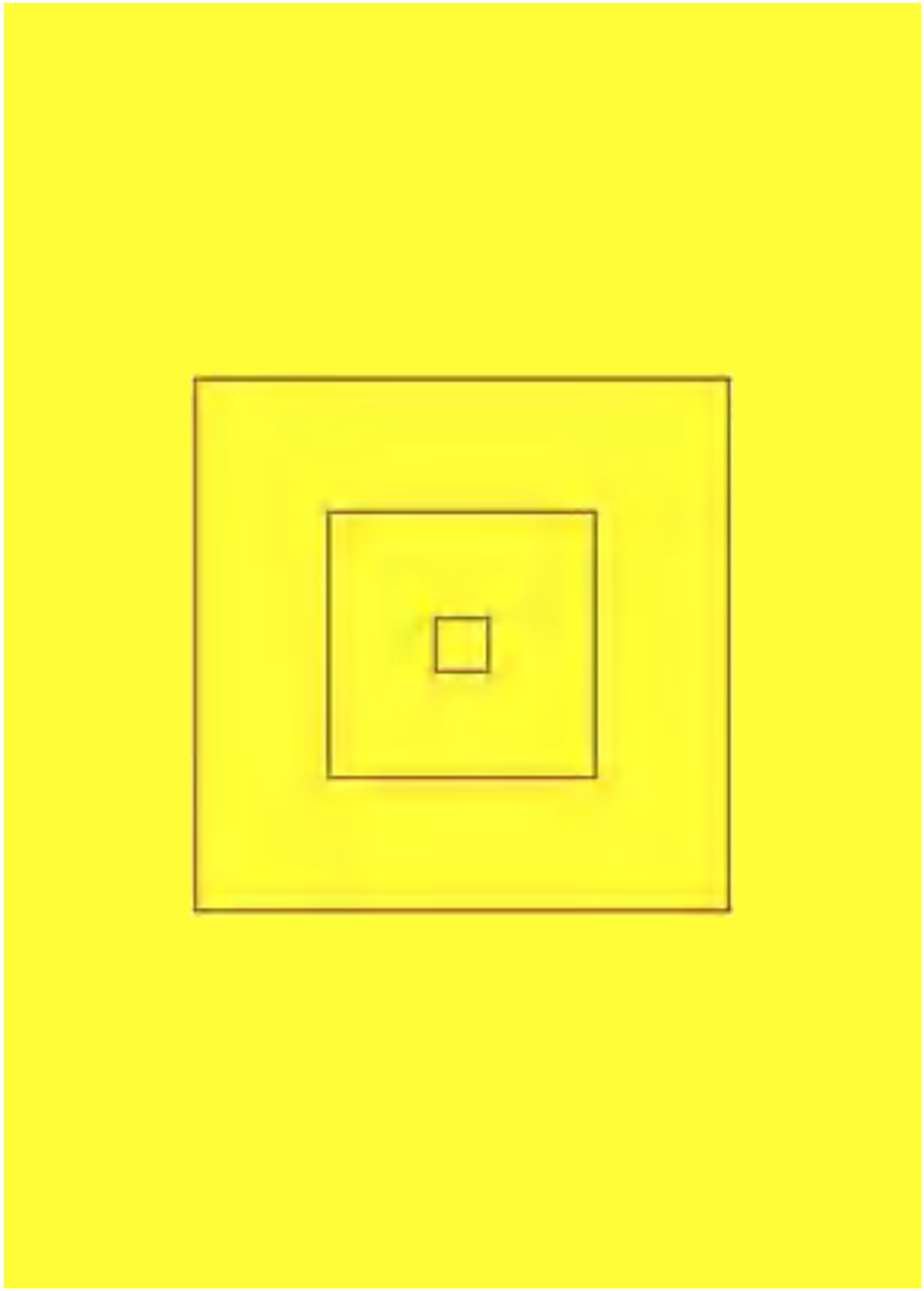
In vielen Einführungen in das Programmieren wird als erstes Beispiel für eine einfache Befehlsabfolge der Satz *Hallo Welt* auf dem Bildschirm ausgegeben[6]. Für mich ist das Zeichnen eines Quadrats gewissermaßen das *Hallo Welt* in den Sprachen der Logo-Familie! Am Quadrat können nämlich gleich mehrere wichtige Konzepte veranschaulicht werden:

Wiederholungen: Im **wiederhole**-Block werden dessen Befehle so oft ausgeführt, wie angegeben, beim Quadrat also viermal.



Erweiterbarkeit und Wiederverwendbarkeit: Die Schildkrötensprache kann durch eigene **Prozeduren** erweitert werden. Diese Befehlsfolgen können dann immer wieder gebraucht werden. In der Prozedur **quadrat** werden die Befehle zum Zeichnen der vier gleichlangen Seiten und das jeweilige Drehen der Schildkröte um 90 Grad zusammen gefasst.

[6] siehe dazu bei Wikipedia die Liste von *Hallo-Welt*-Programmen in Höheren Programmiersprachen.

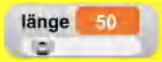


Quadrate (II)

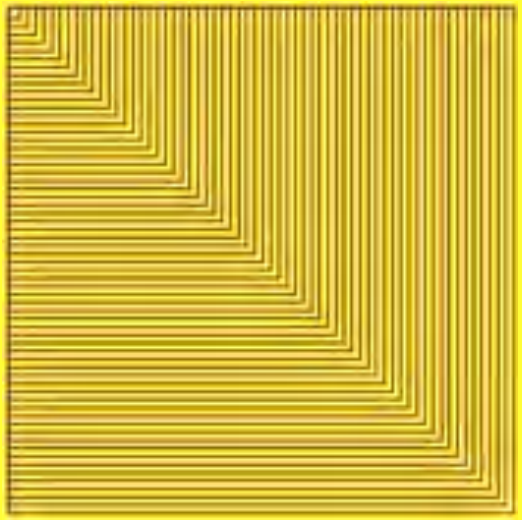
Variablen: Alle möglichen Daten (Zahlen, Texte, Bilder usw.) können in Variablen abgelegt werden. Der Computer findet die Daten sicher wieder, kann sie im Programm immer wieder verwenden und auch verändern.

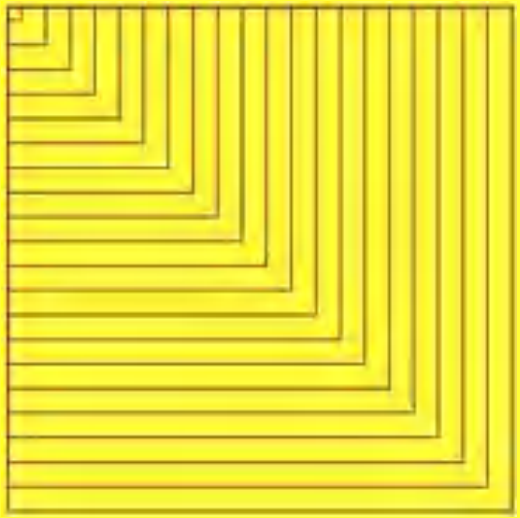
Der Prozedur **quadrat** wird die Seitenlänge als **Eingabeparameter** **länge** übergeben Die Quadrate können so in beliebiger gewünschter Größe gezeichnet werden.

Vereinfachung durch Wiederholungen: Die Wiederverwendung von Prozeduren in Wiederholungsschleifen erlaubt auch komplexere Figuren; so im Folgenden bei **geschachtelte Quadrate** und **Diamanten**, gesteuert über **n_quadrate**, **delta** und **winkel**.

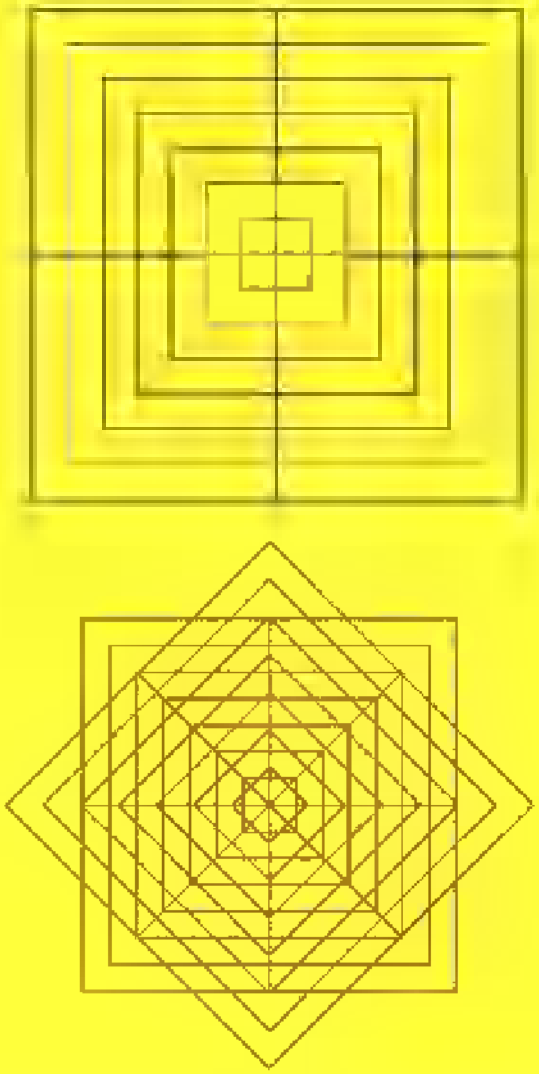


geschachtelte Quadrate

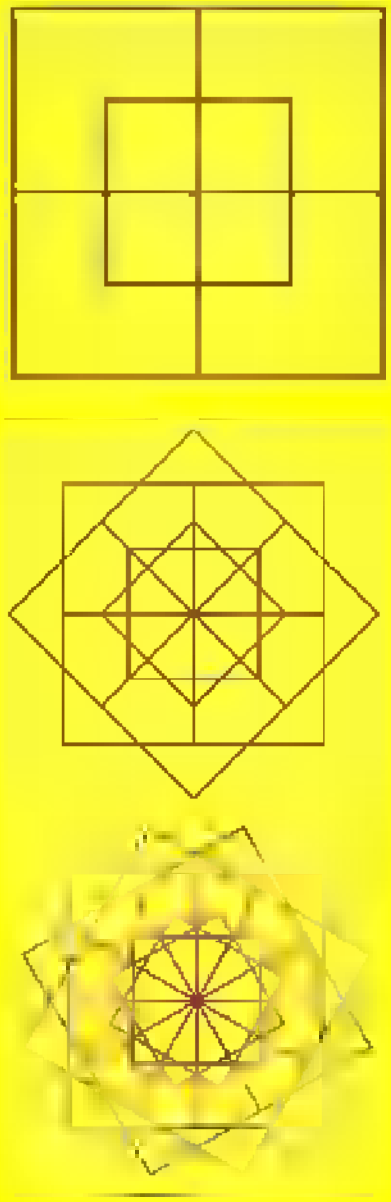


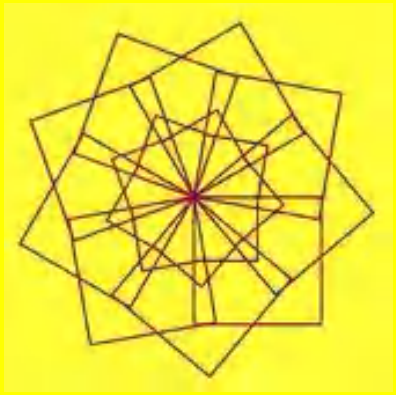


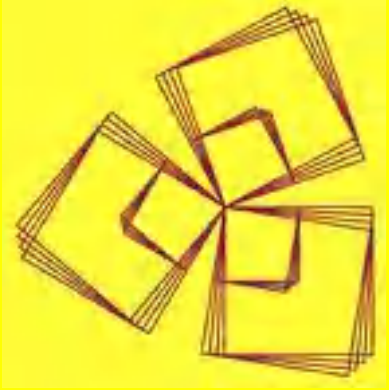
Diamanten

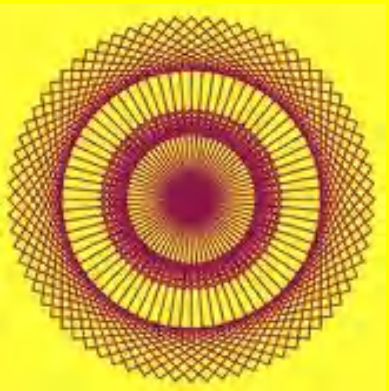


Quadrate

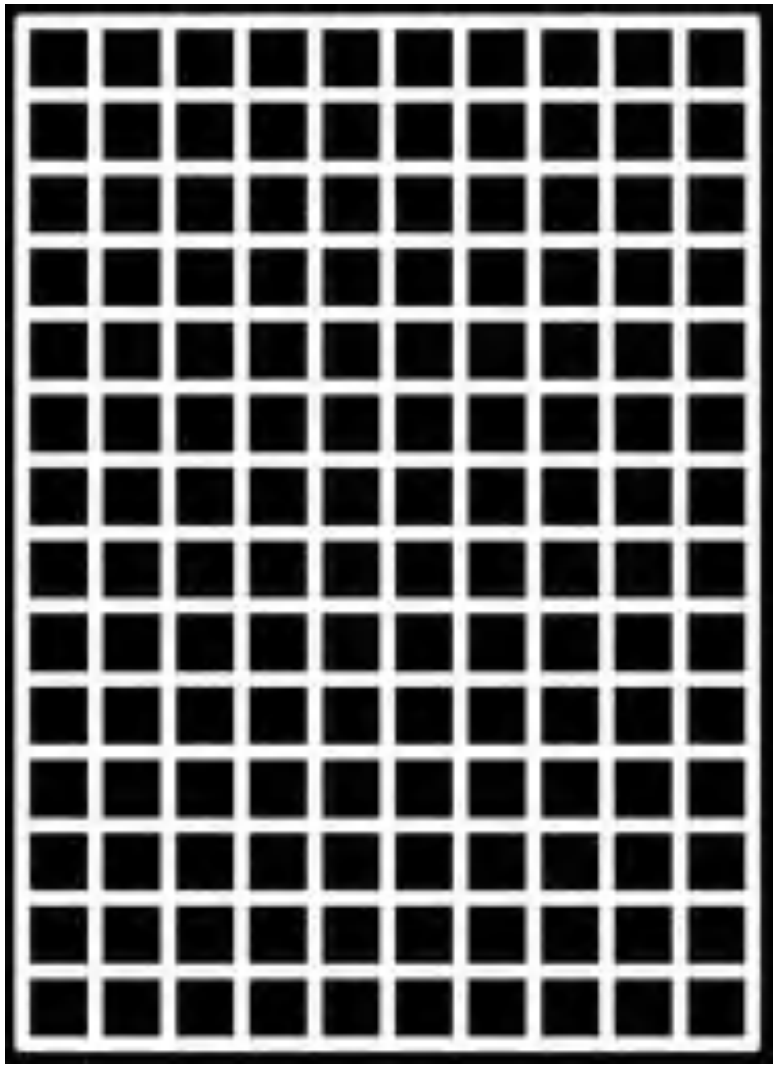




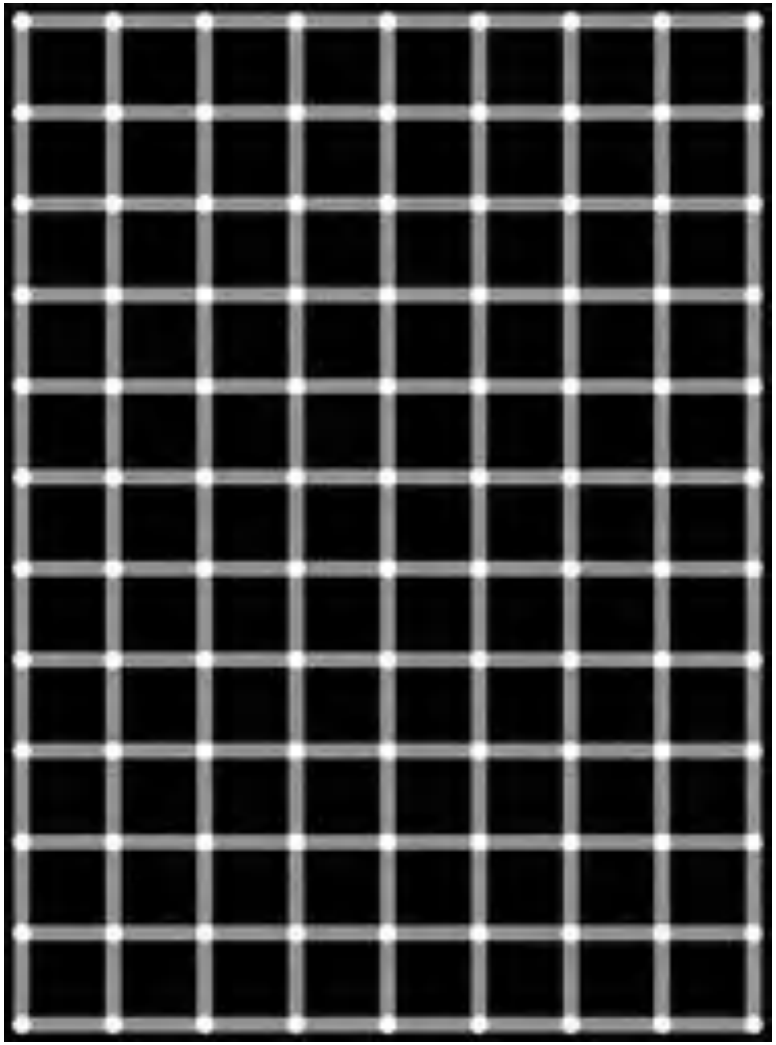




Opticals: Hermann-Hering Gitter (graue Punkte)



Opticals: Hermann-Hering Gitter (springende Punkte)

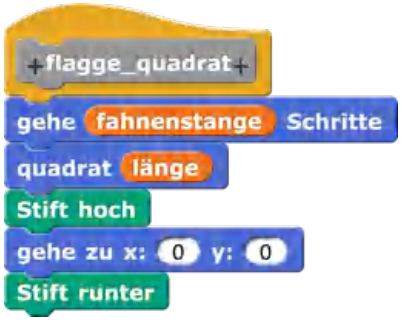


Flaggen, Spinnen, Wirbel

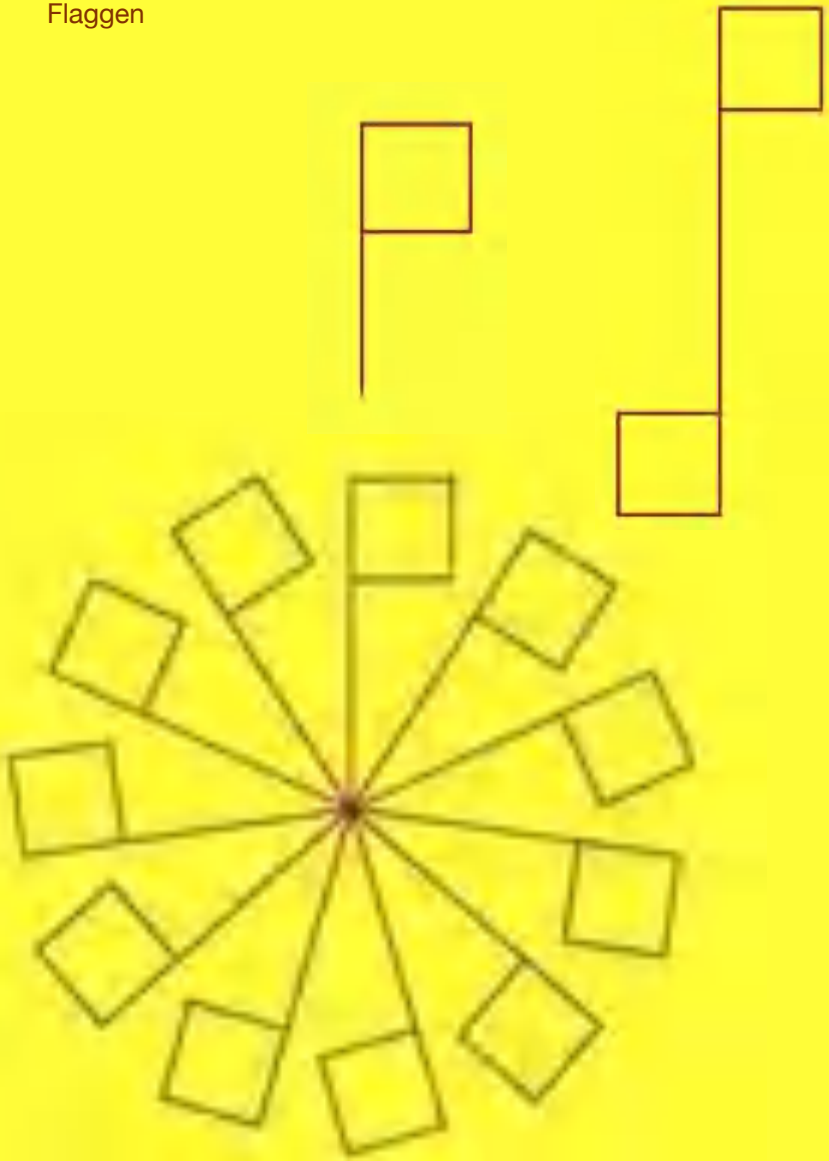
Wiederverwendbarkeit und Ortsneutralität: Der Vorteil von Prozeduren ist ihre Wiederverwendbarkeit in neuen Kontexten. Das gilt natürlich auch für die Prozedur **quadrat**. Wichtig ist dabei die Ortsneutralität, d.h. die Rückführung der Schildkröte an ihren Ausgangspunkt mit ihrer ursprünglichen Blickrichtung.

Eine Prozedur **flagge_quadrat** zeichnet die Fahnenstange der Länge **fahnenstange** und dann das Quadrat mit der Seitenlänge **länge**. Abschließend wird die Schildkröte immer in die Mitte der Bühne geführt.

So lassen sich aus einfachsten Strukturen leicht komplexere Strukturen zusammensetzen. Das gilt für die Spinnen und Wirbel (auf der folgenden Doppelseite).



Flaggen



Spinne mit Nachwuchs

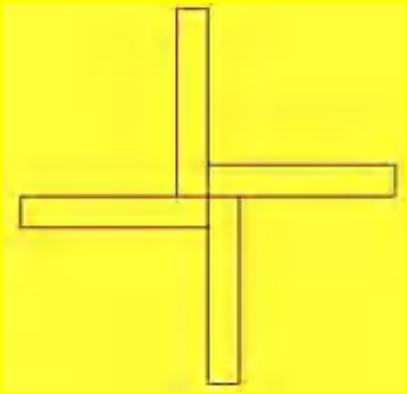
Rechtecke, Parallelogramme

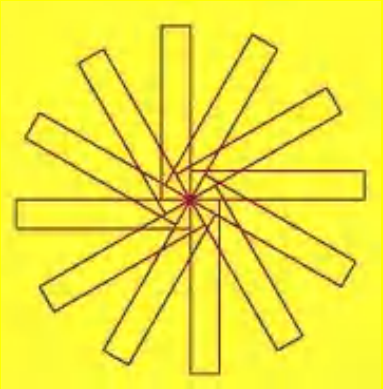
Verändern und Erweitern: Prozeduren können als Ausgangspunkt für neue Verwendungszwecke genommen werden. So kann die Prozedur **quadrat** leicht zu einer Prozedur **rechteck** verändert werden, der die Breite und die Höhe des Rechtecks als Werte **a** bzw. **b** übergeben werden:

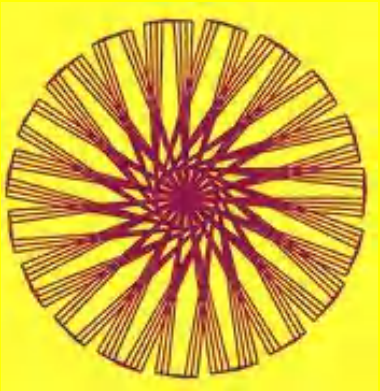
Die Prozedur **rechteck** kann wiederum zu einer Prozedur **parallelogramm** erweitert werden, wenn neben den Seitenlängen der Winkel **beta** übergeben wird (der Winkel **alpha** ergibt sich im Parallelogramm als **180 - beta**).

Die Prozedur **parallelogramm** enthält nun die Ausgangsprozeduren als Sonderfälle, nämlich die Prozedur **rechteck** mit **beta = alpha = 90** und diese die Prozedur **quadrat** mit zusätzlich **a = b**!

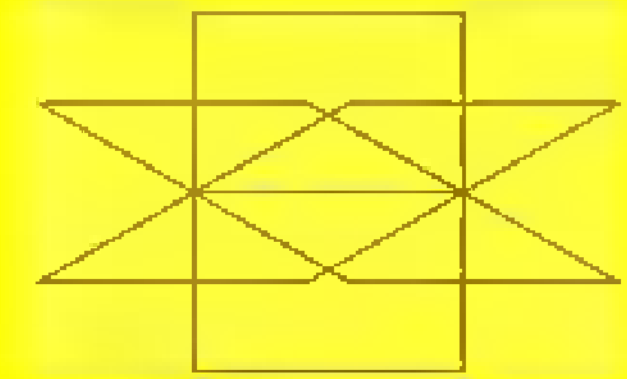
Rechtecke

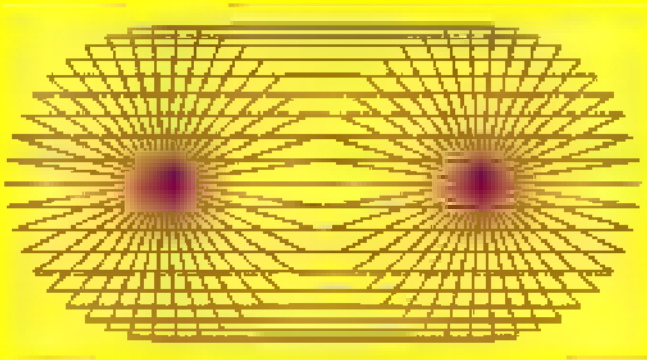


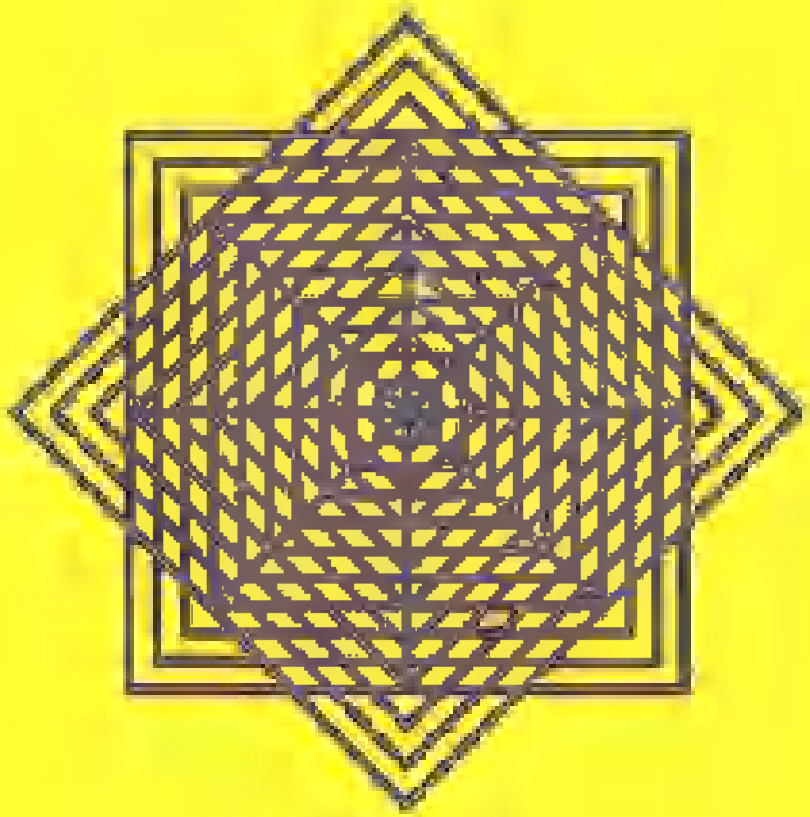


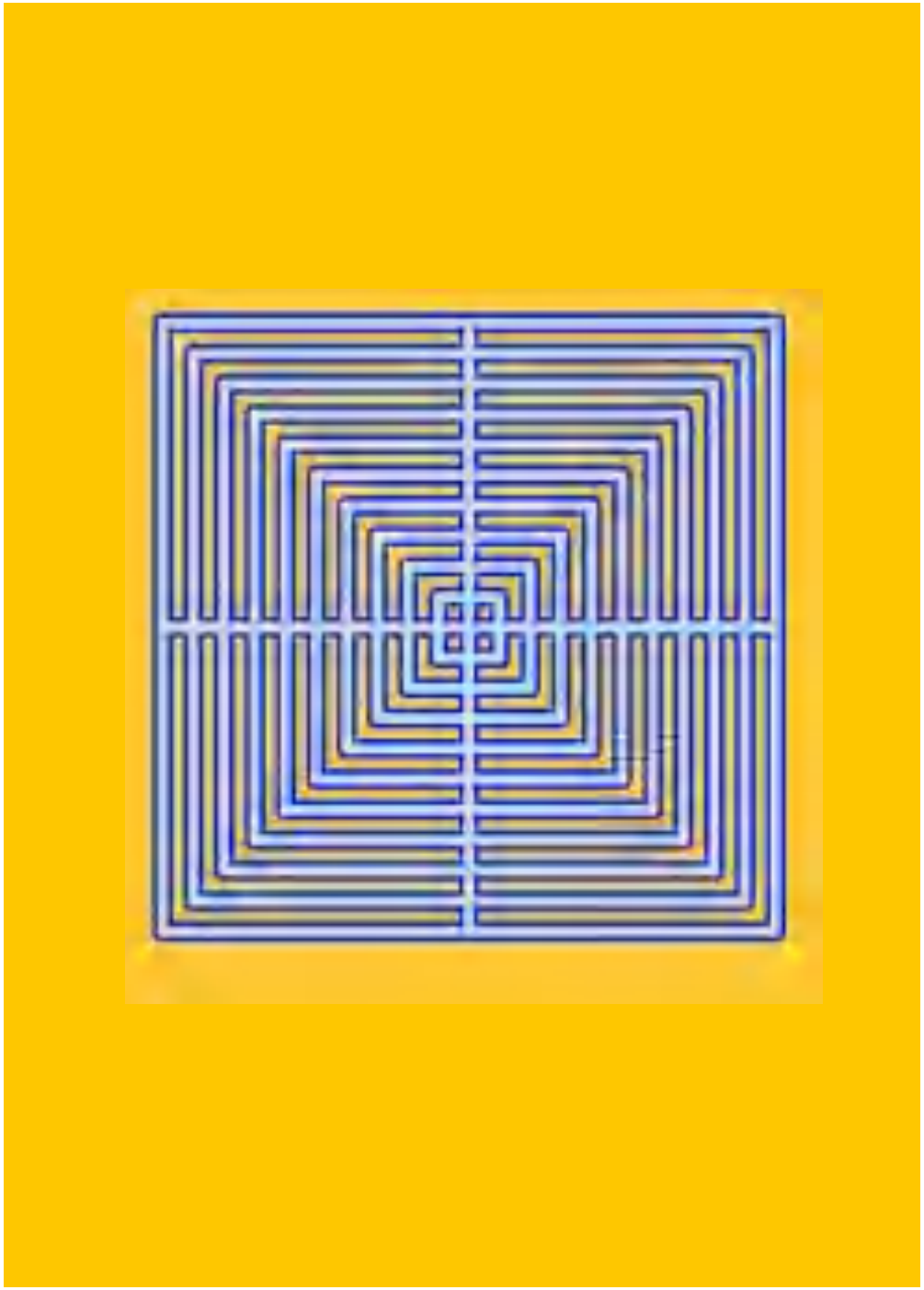


Parallelogramme

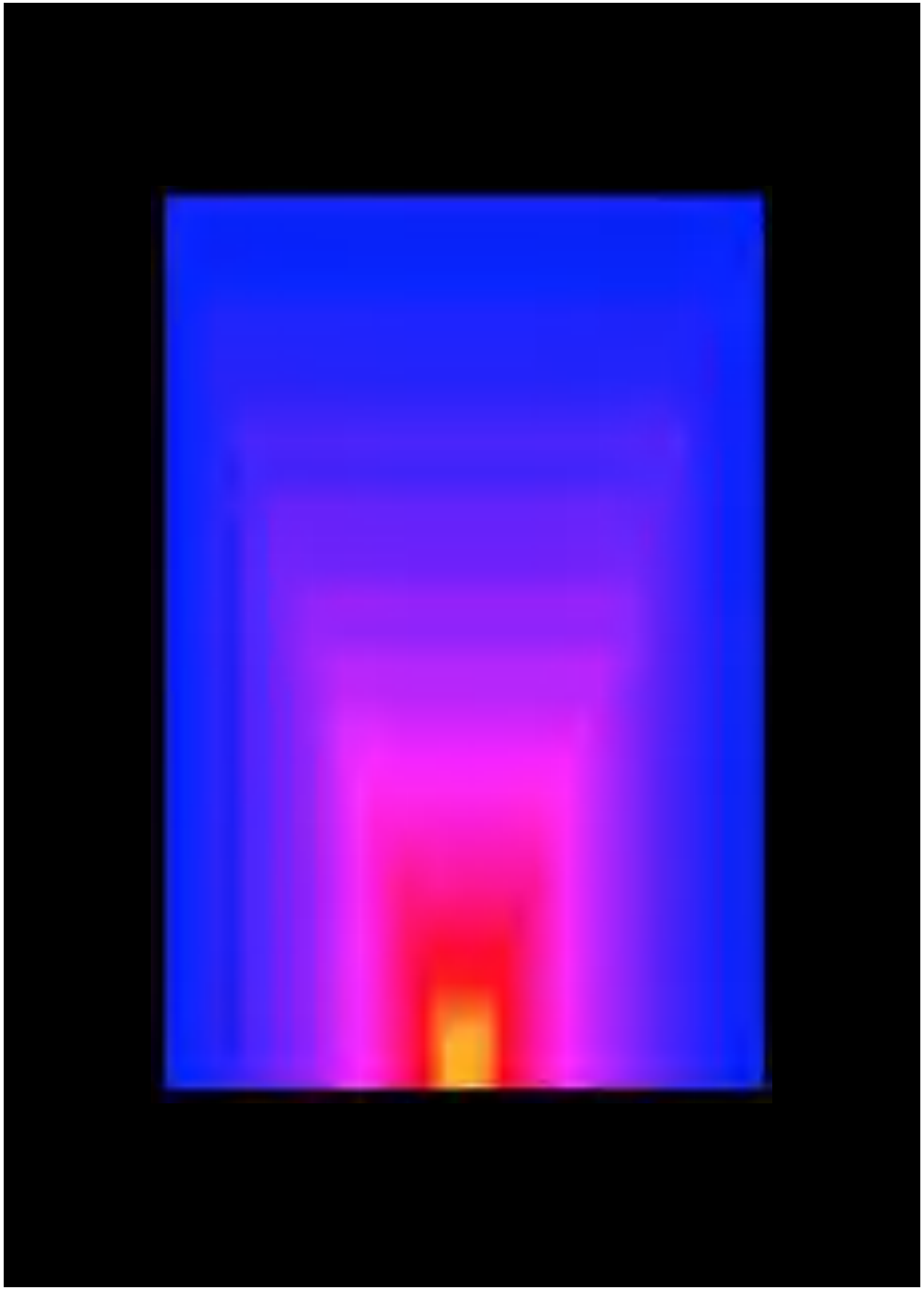








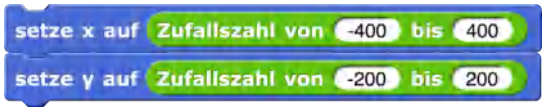
Opticals: Raumwirkung durch Farbtöne



Zufall

In den bisherigen Bildern war alles eindeutig und damit vorhersehbar durch die Programmbefehle festgelegt. Für Überraschungsmomente in den Bildern kann der **Zufall** sorgen, denn alle Eigenschaften der Schildkröte können leicht zufällig geändert werden. Grundlage ist die **Zufallsfunktion** **Zufallszahl von 1 bis 10**, die **gleichverteilte** Zufallszahlen zwischen 1 und 10 liefert.

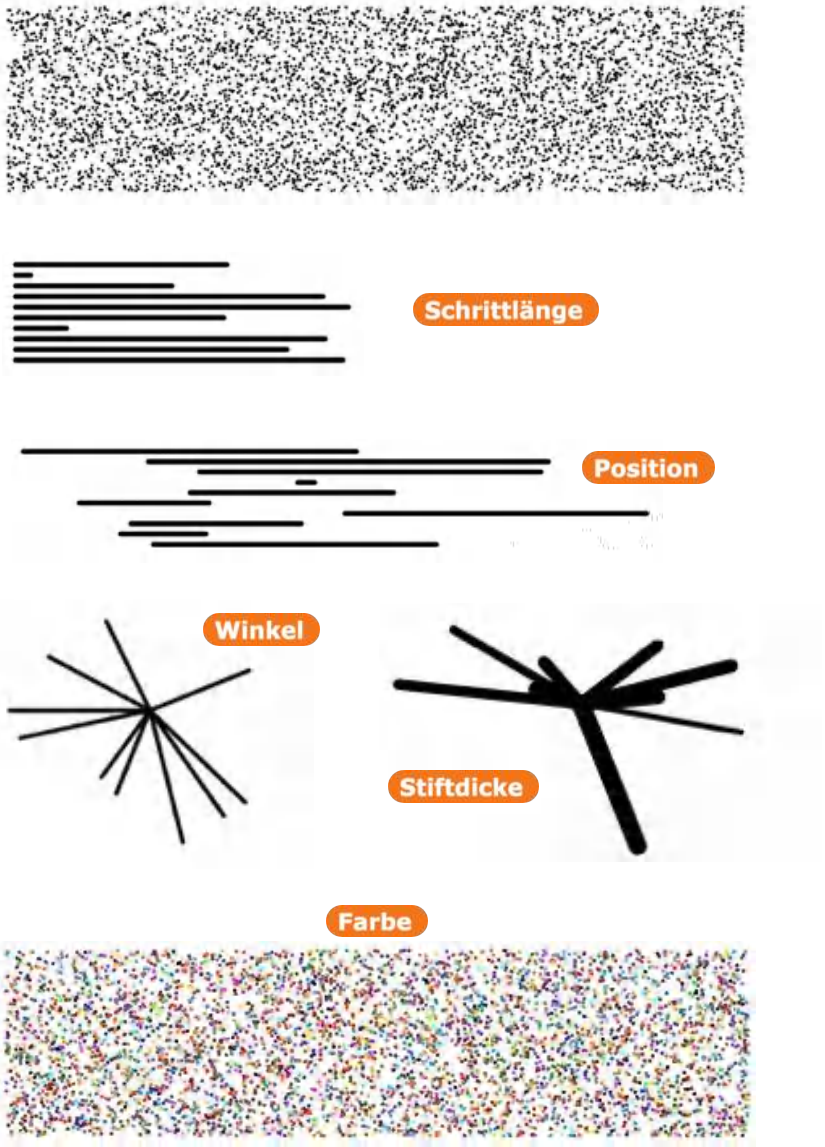
Der Bereich für die Zufallszahlen kann frei gewählt werden. Bei solchen Begrenzungen wird vom **gelenkten Zufall** gsprochen. So werden zum Beispiel mit



zufällige Positionen der Schildkröte festgelegt. Das Bild rechts oben zeigt entsprechend die ziemlich gleichmäßige Verteilung von Punkten.

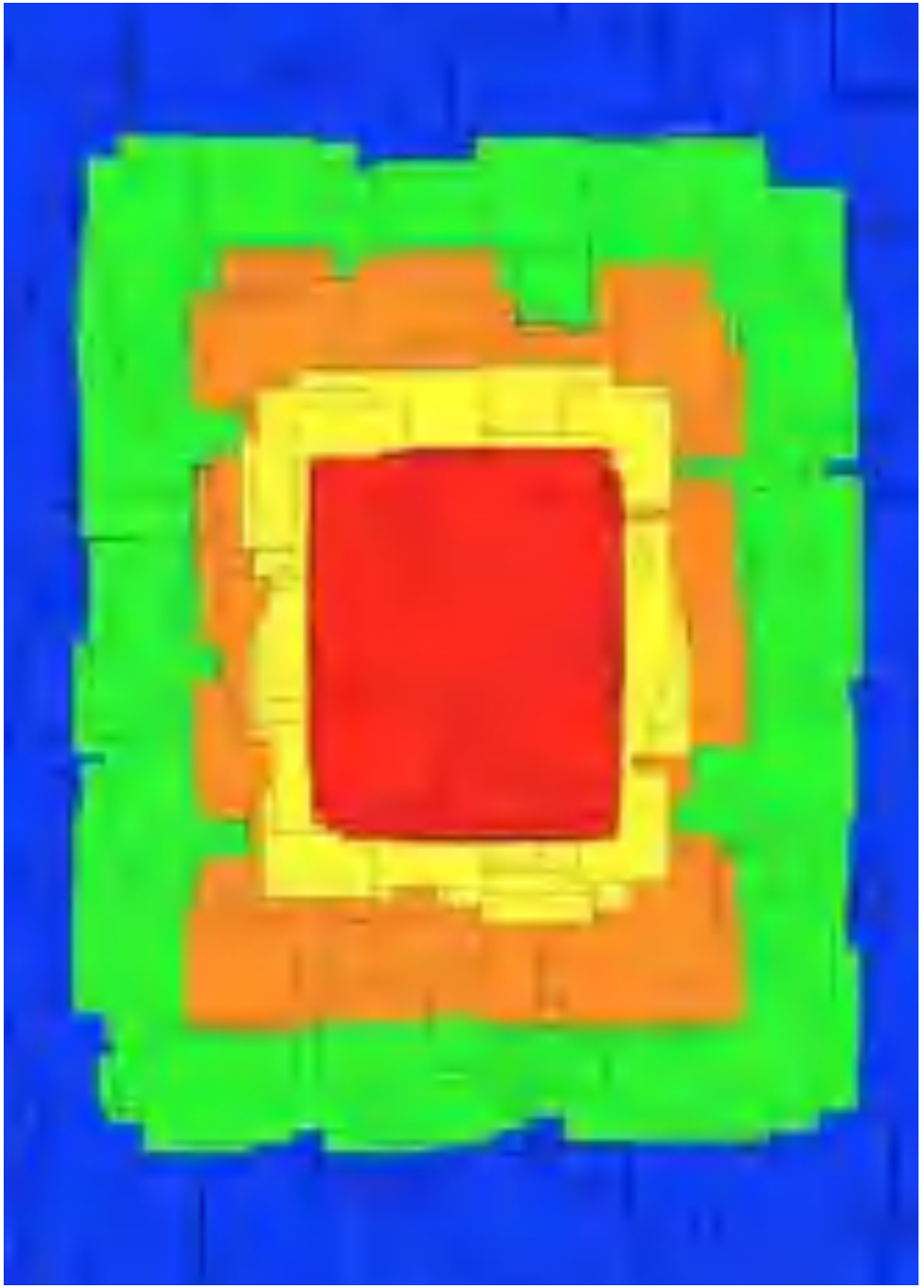
Der Zufall kann alles ändern! Das betrifft unter anderem

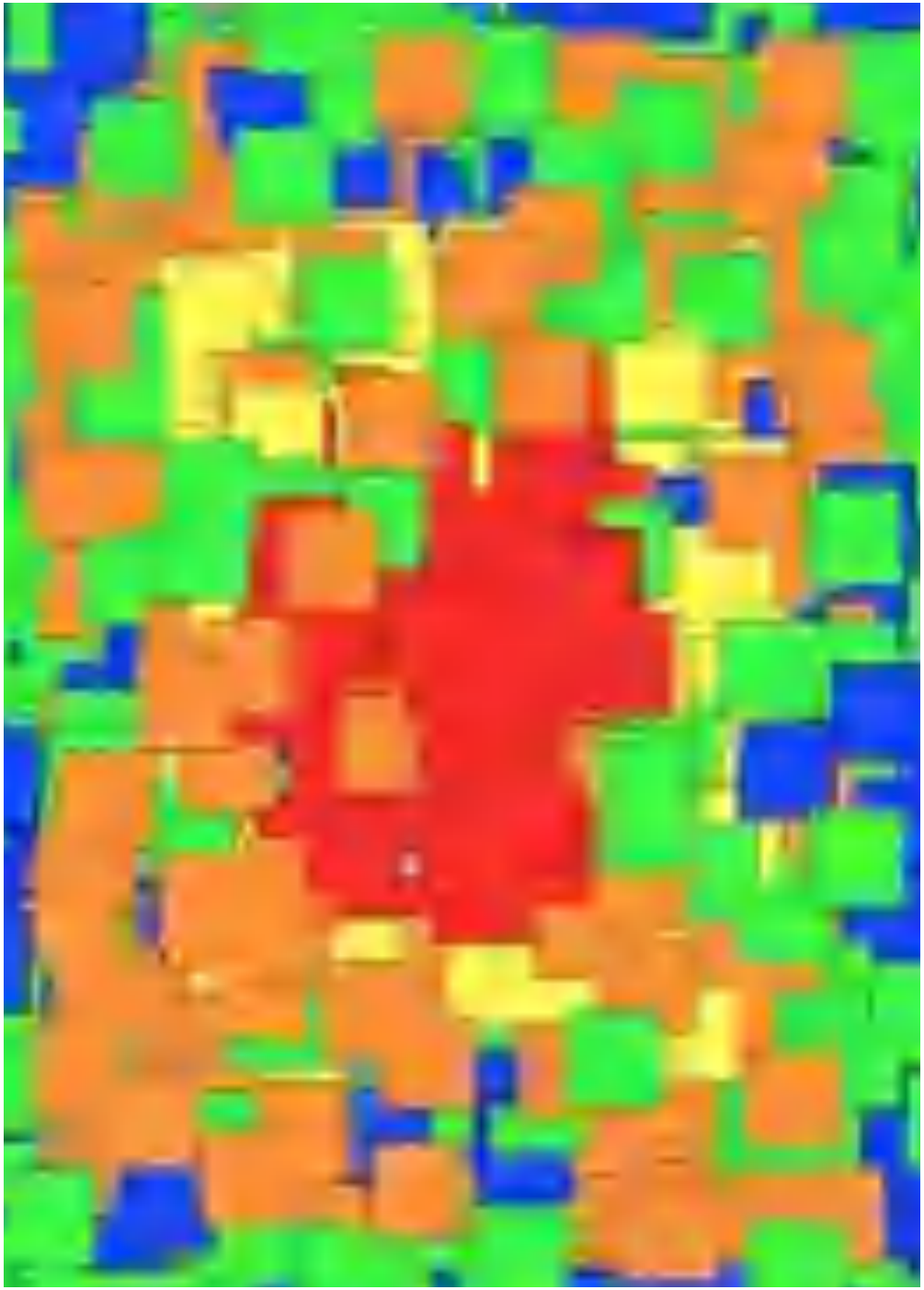
- die **Schrittlänge** mit **gehe**,
- die **Position** mit **gehe zu**,
- den **Winkel** mit **drehe**,
- die **Stiftdicke** mit **sette stiftdicke auf**,
- oder die **Farbe** mit **setze stiftfarbe auf**.



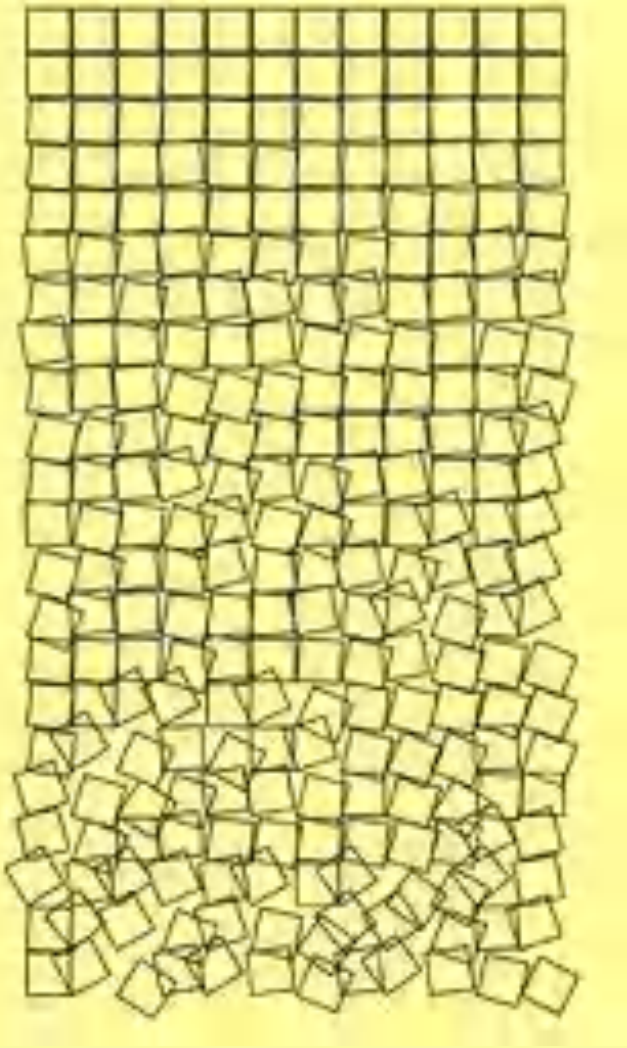
Hommage à Nees: Linienbüschel

Hommage à Nees: Eckenlinien

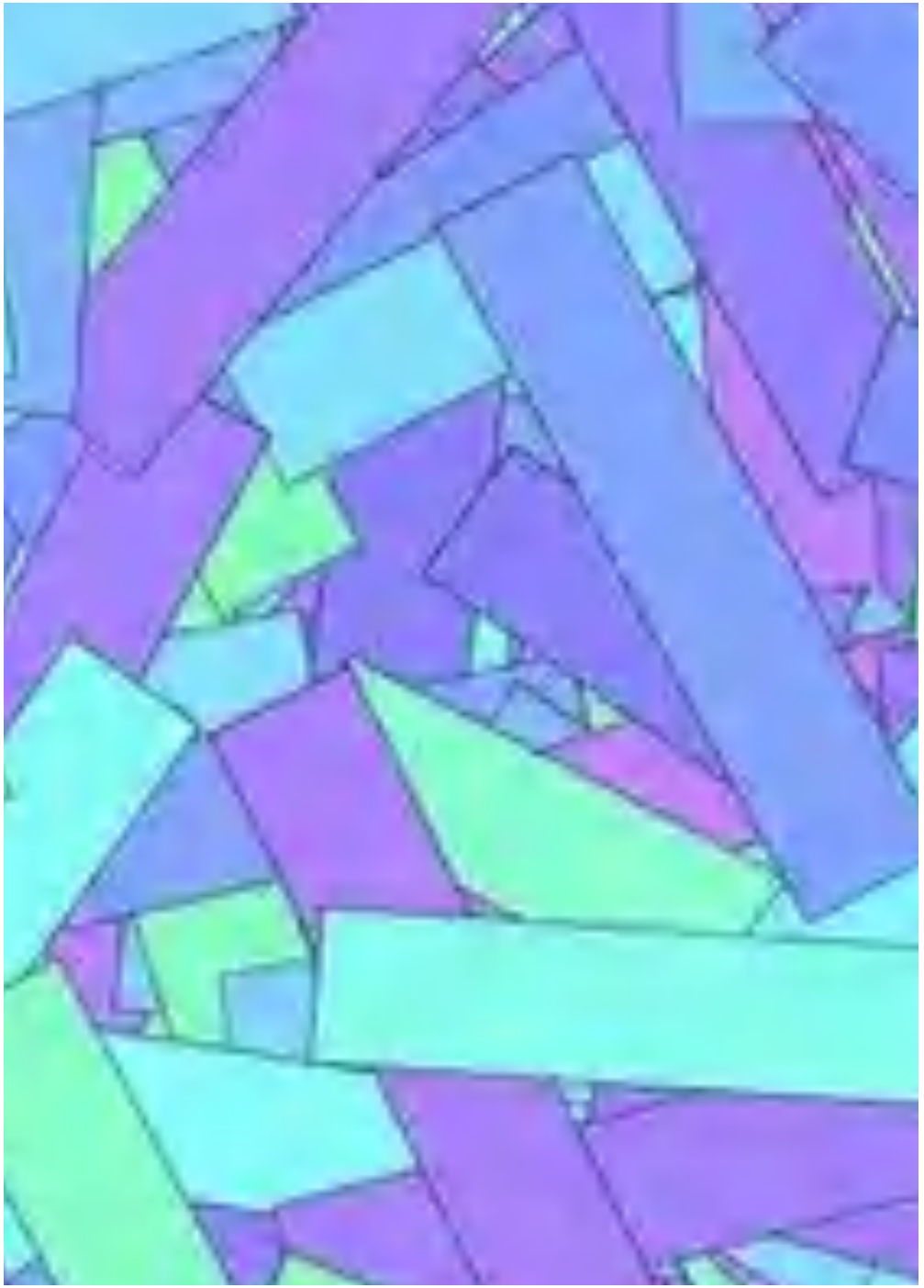




Hommage à Franke: Quadrate



Hommage à Nees: Schotter



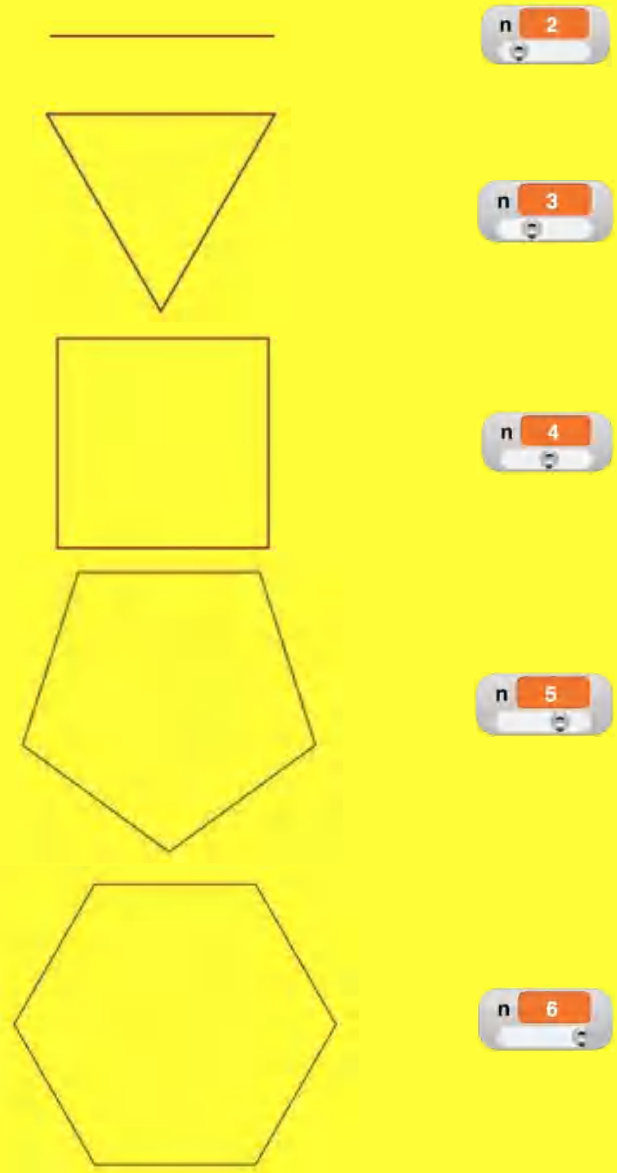
Polygone

Das **Verallgemeinern** von Programmen bzw. Prozeduren ist **der Schritt vom Vertrauten zum Unbekannten**. Eine kleine Veränderung der Prozedur **quadrat** macht das deutlich. Allein mit der Einführung eines Parameters **n**, mit dem die Zahl der Ecken eines Vielecks bestimmt wird, können wir in einer neuen Prozedur **polygon** statt der Quadrate beliebige regelmäßige Vielecke (Polygone) mit jeweils gleichlangen Seiten und gleichen Winkeln zeichnen.

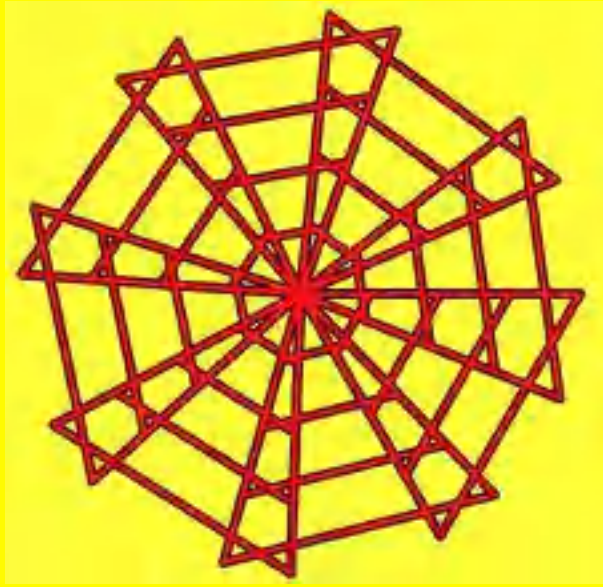
Gegenüber der Prozedur **quadrat** ändert sich neben den Eingabewerten nur der Drehwinkel. Mit **360/n** wird sicher gestellt, dass die Winkelsumme im Polygon immer 360 Grad ergibt.

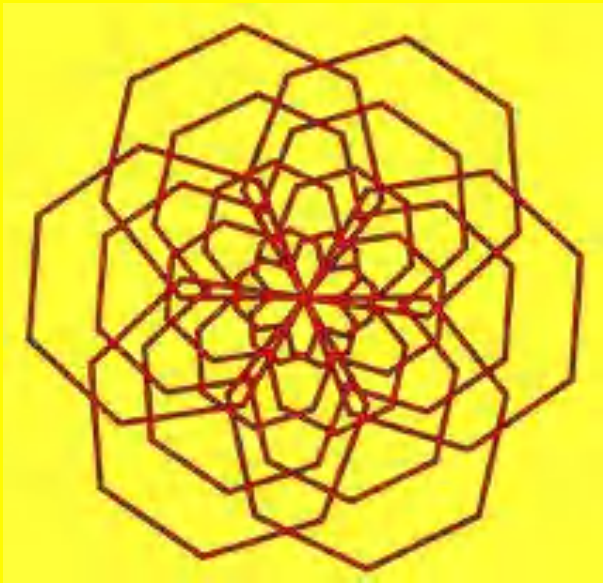
Interessant: Bei allen regelmäßigen Vielecken dreht sich die Schildkröte jeweils um 360 Grad. Papert (Papert, 1982, S. 107) nennt das den **Vollständige-Schildkrötenreise-Satz**:

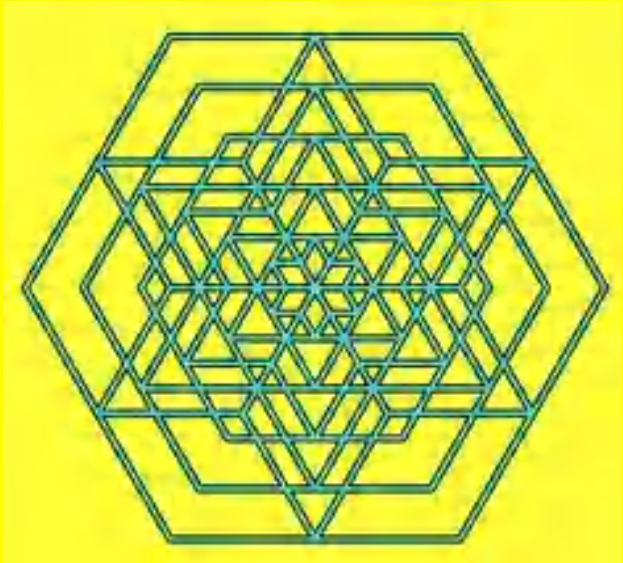
„Wenn die Schildkröte eine Reise um die Grenzen einer Fläche macht und in demselben Zustand aufhört, in dem sie angefangen hat, dann ist die Summe aller Drehungen 360 Grad.“

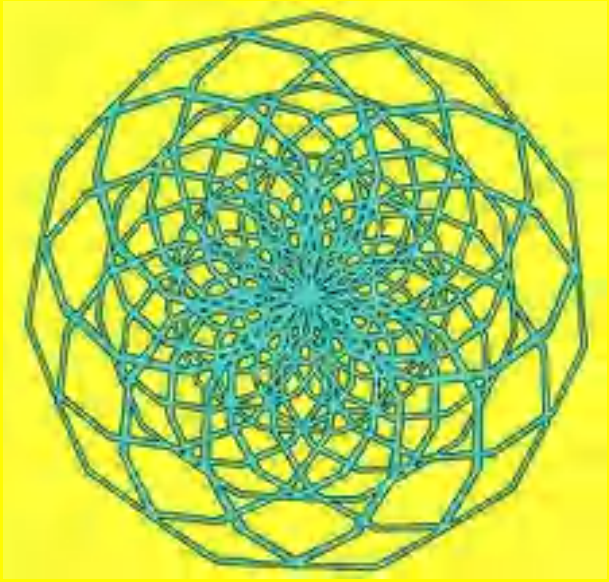


geschachtelte Polygone

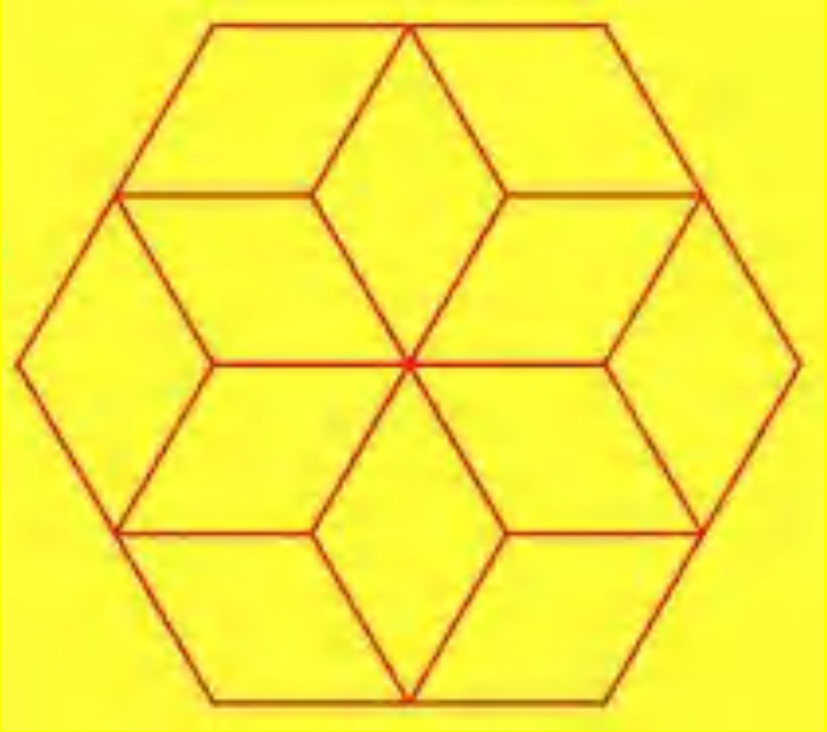


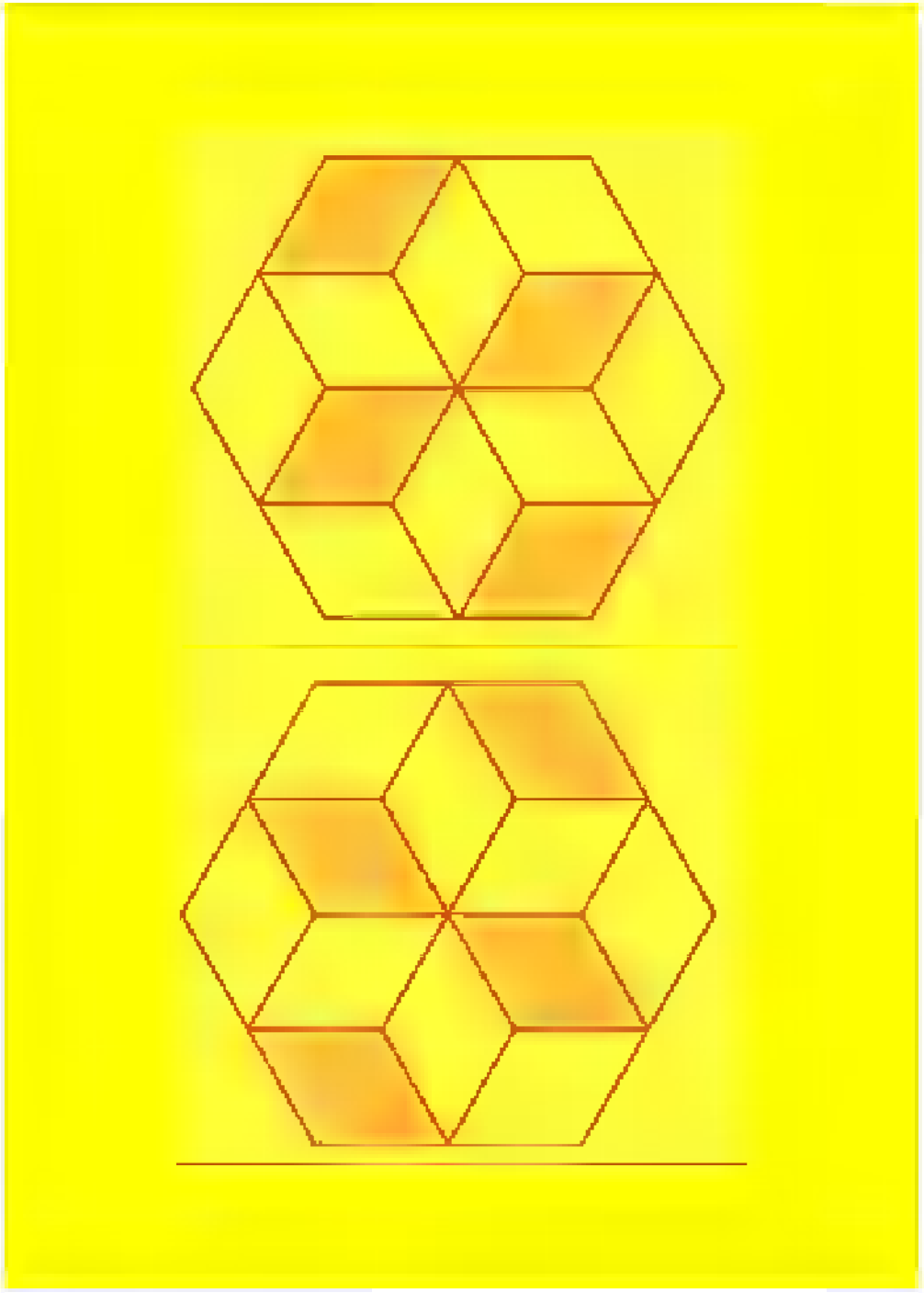


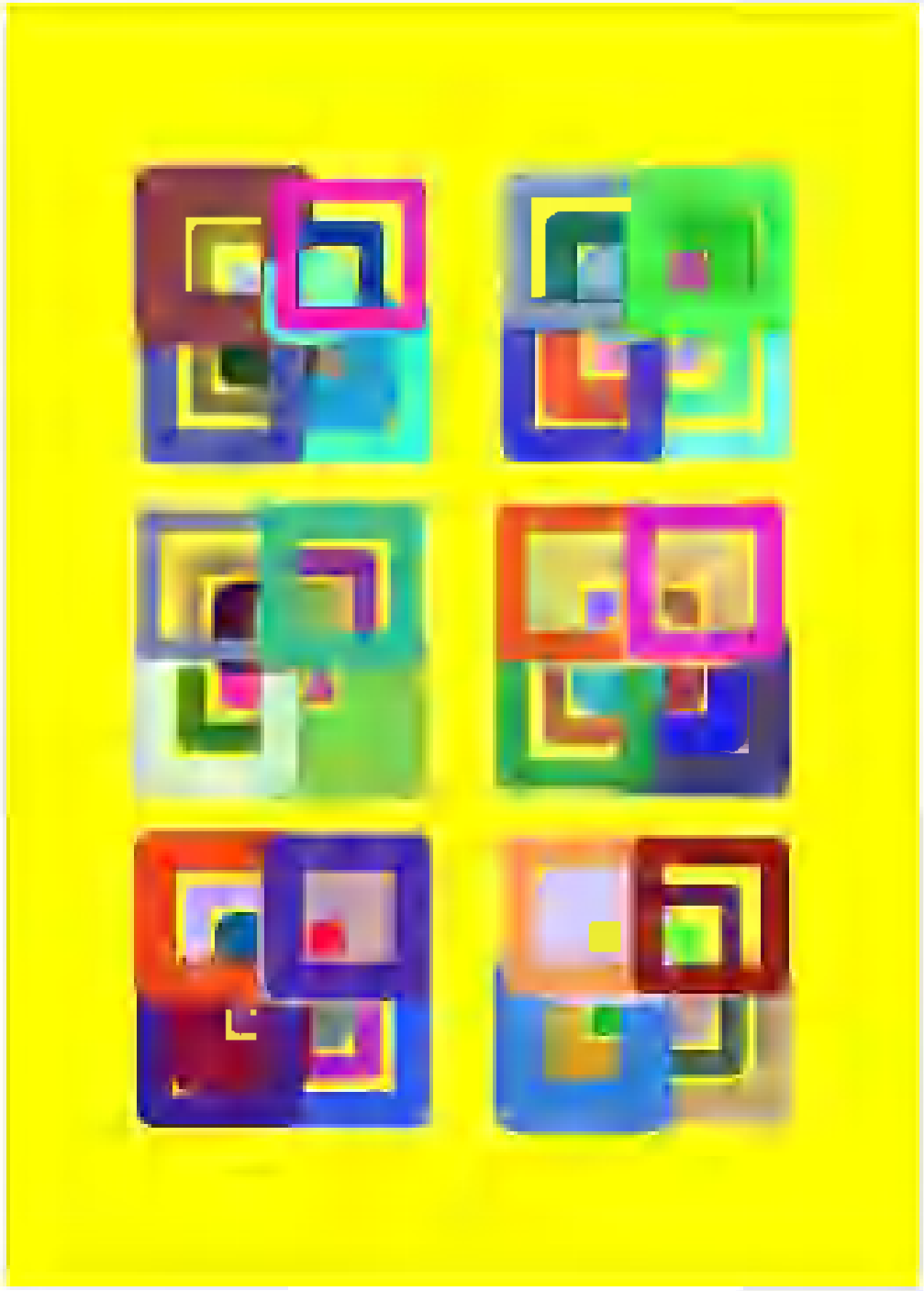


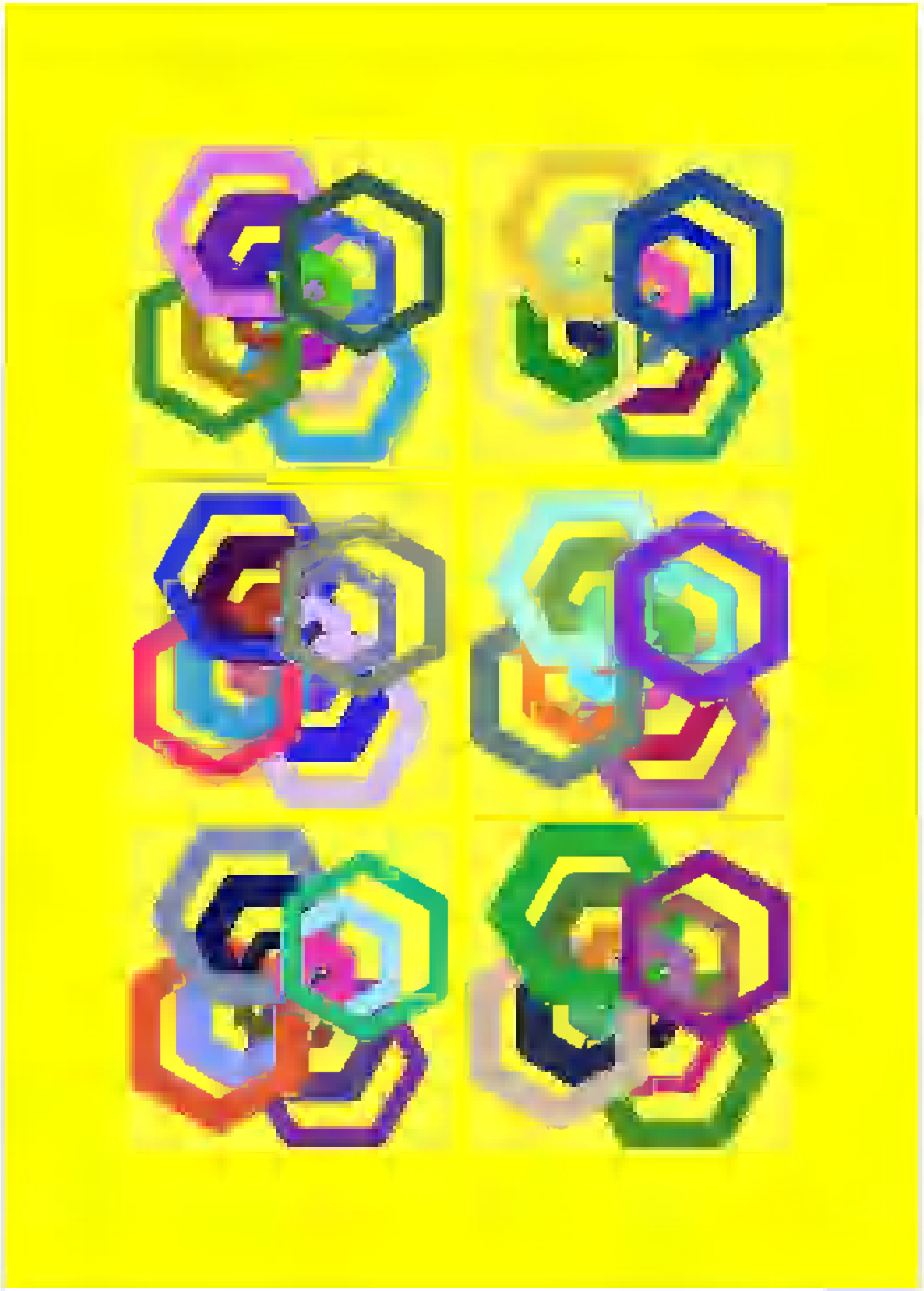


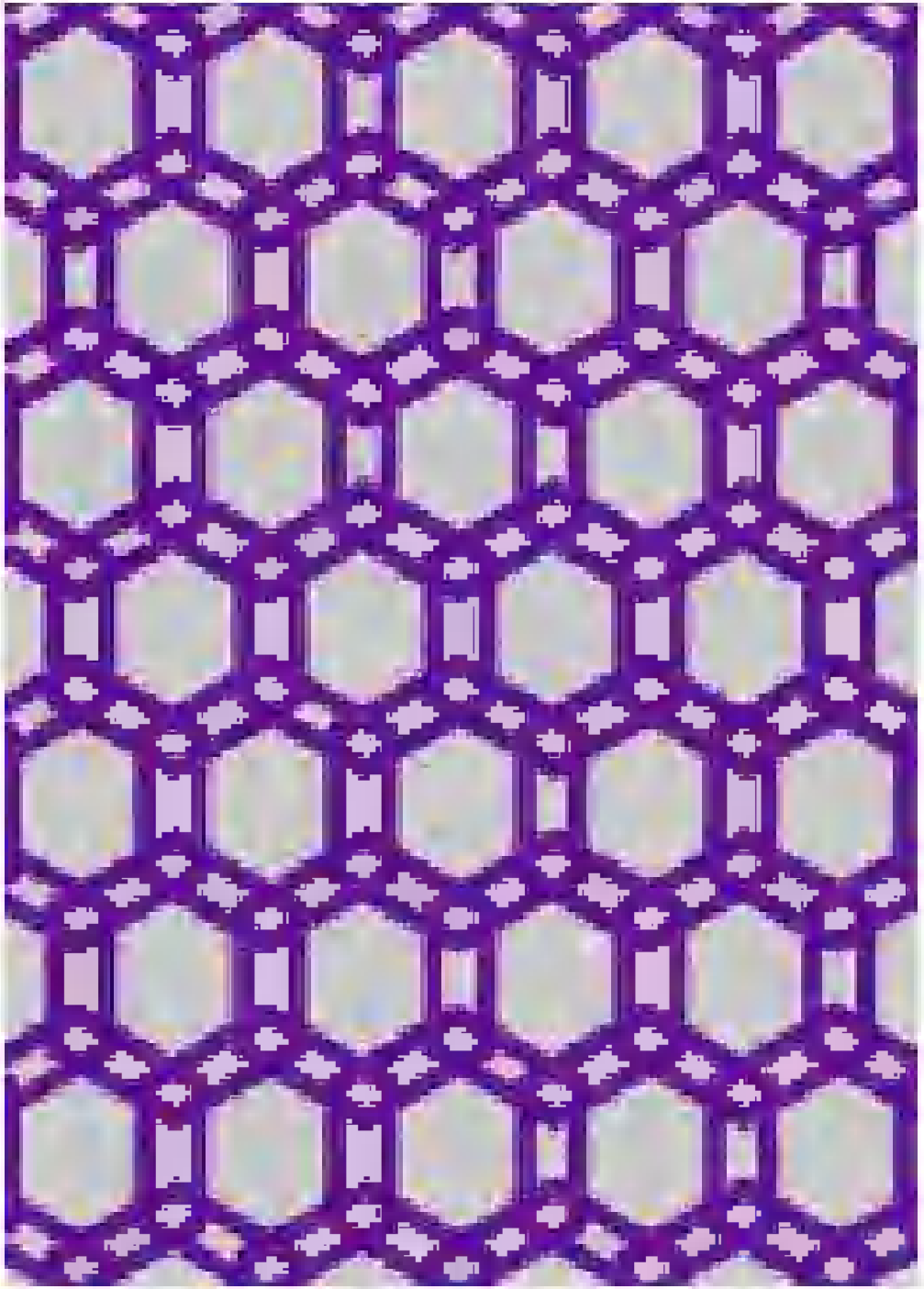
Opticals: Kippfigur Necker-Würfel
(6 Sechsecke, Winkel 60)

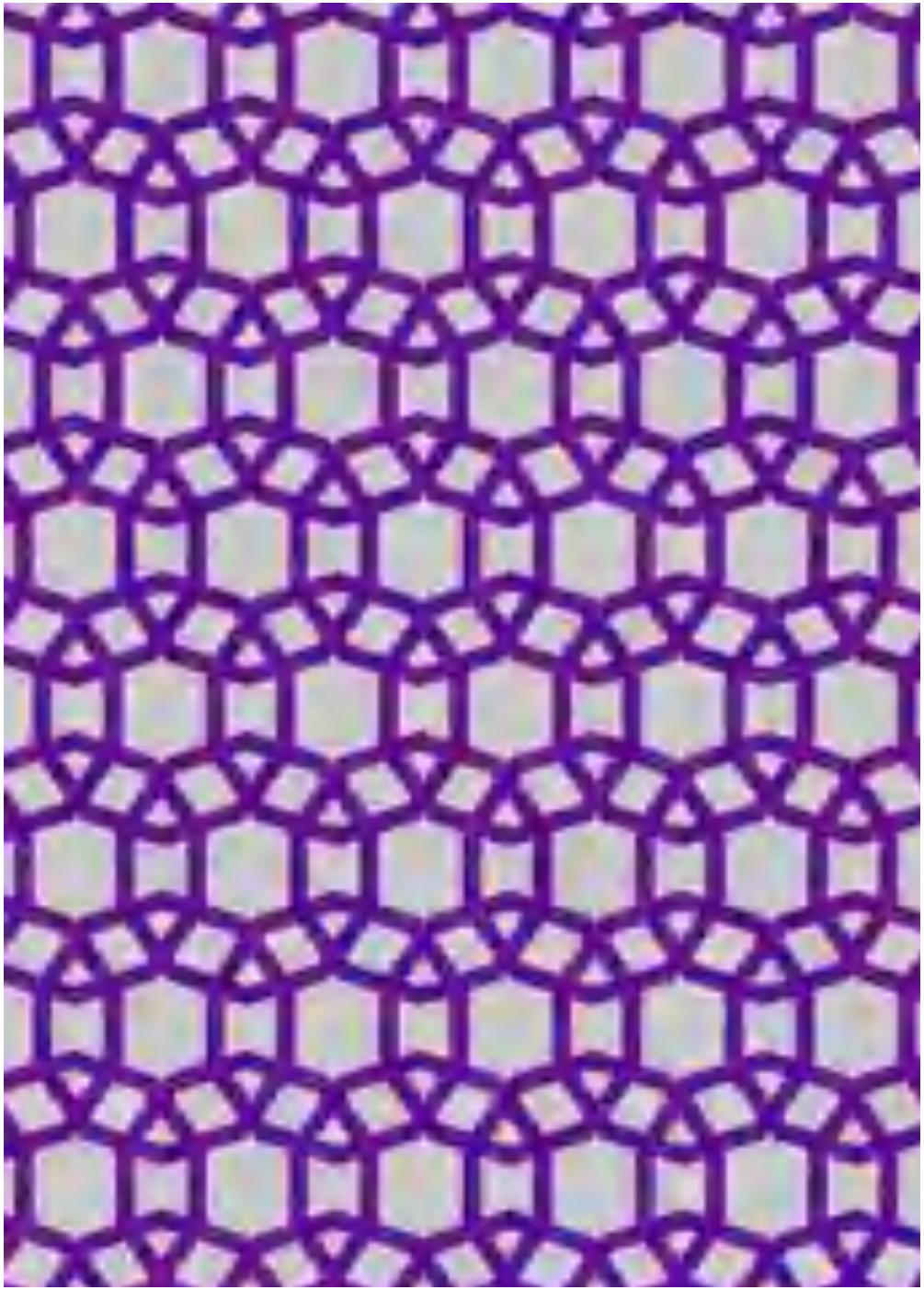


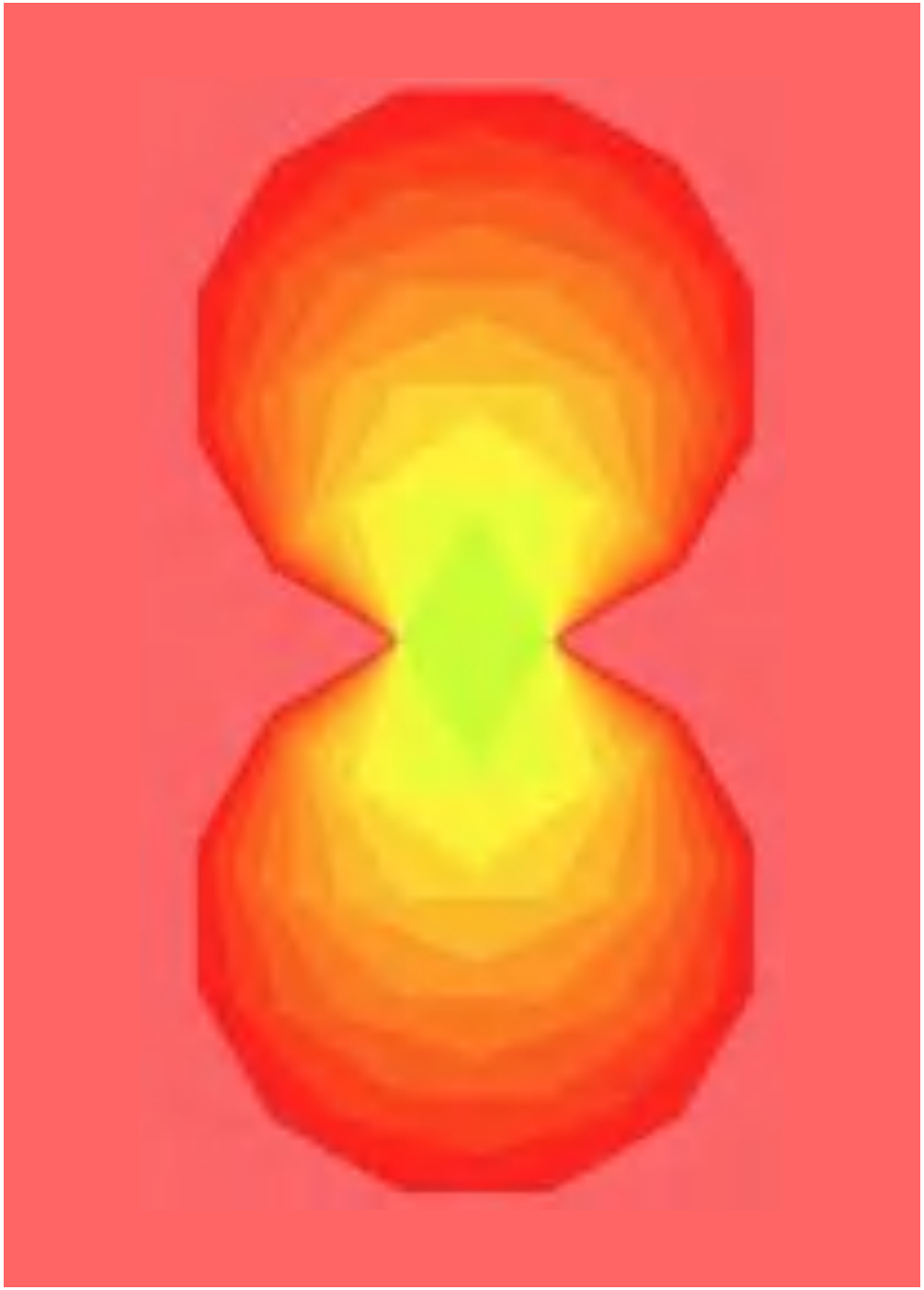


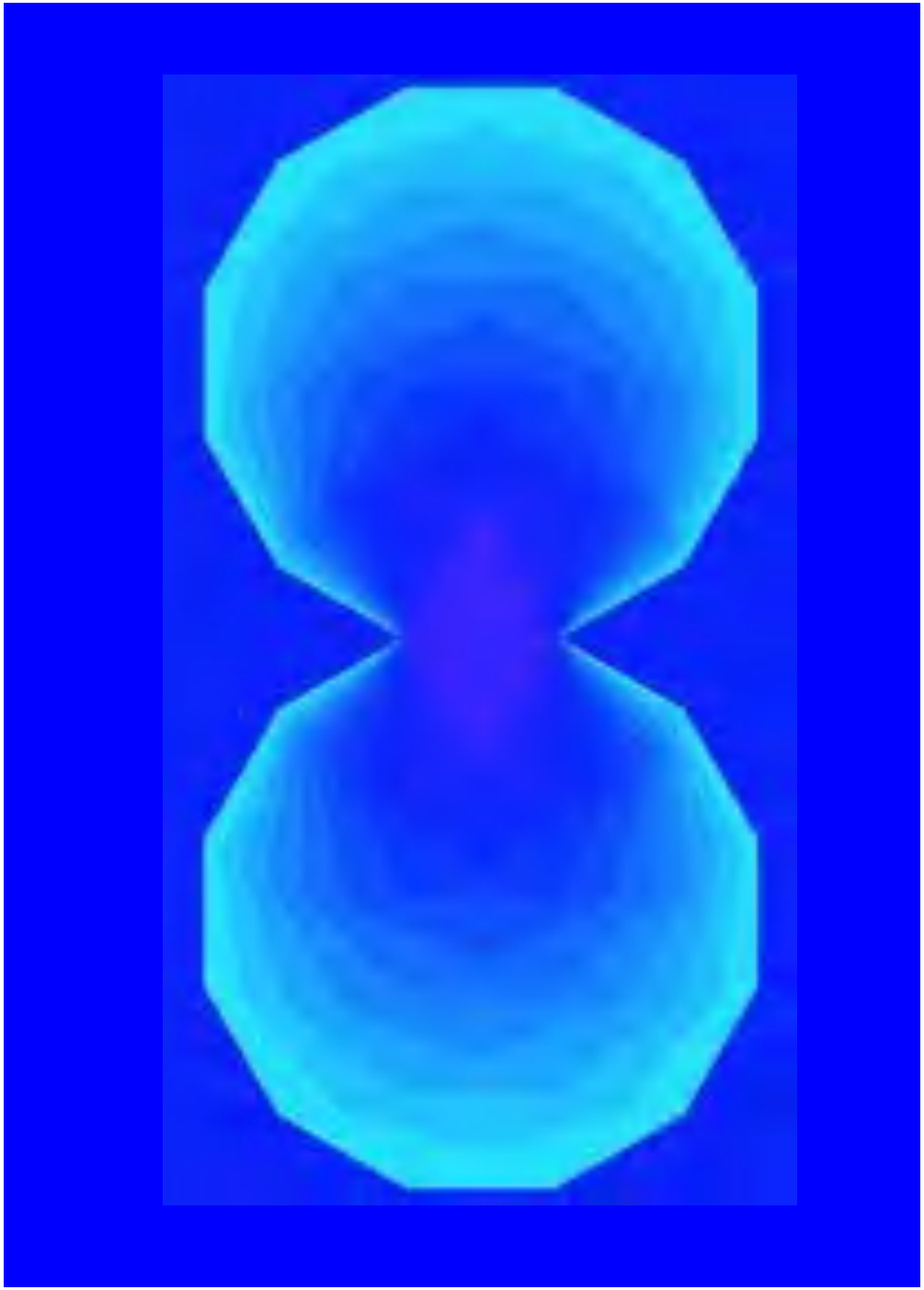












Squiggle, Squaggle & Co.

Für den **Vollständige-Schildkrötenreise-Satz** gibt es einen wichtige Verallgemeinerung. Auch bei offenen Polygonzügen endet die Schildkröte im Ausgangszustand, wenn die Summe aller Rechts- und Linksdrehungen wieder 360 Grad oder ein Vielfaches davon ergibt.

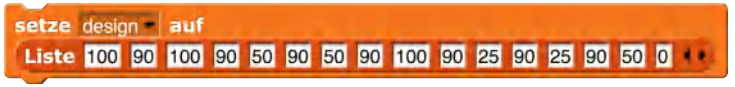
Ein Beispiel mit dem lustigen Namen **squiggle** hat Brian Harvey eingeführt (Harvey, 1982, S. 186 ff.). Die dabei immer gleiche Abfolge paarweiser Vorwärts- und Drehbewegungen lässt sich einfach und flexibel mit einer Prozedur **designs** und mit der Zusammenfassung der Bewegungsdaten in einer **Liste** umsetzen.

Listen sind Felder, in denen Zahlen und/oder Texte, Listen und sogar Programmcode abgespeichert und wieder abgerufen werden können.

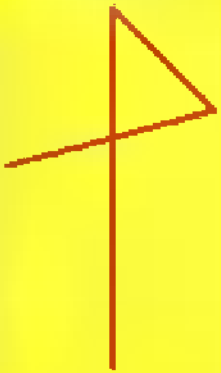
Die Bewegungsdaten für **squiggle** und zwei weitere Varianten namens **squaggle** und **squoggle** ergeben dann folgende Listen:

Mit **designs** lassen sich nun gleichartige Varianten leicht zeichnen. Aber erst durch Wiederholungen und durch Ändern der Werte für die Bewegungen entstehen die eigentlich interessanten Muster.

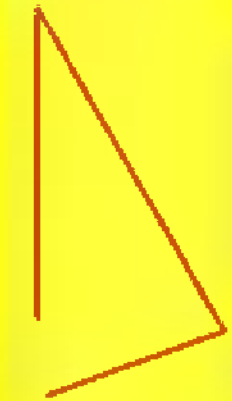
Von Hal Abelson (1983, S. 49) gibt es ein ähnliches Beispiel mit dem schlichten Namen **design**.



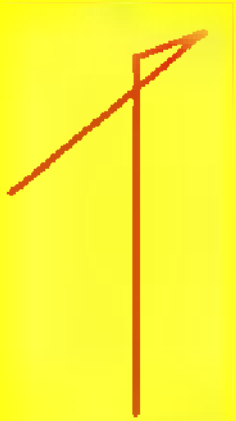
squiggle



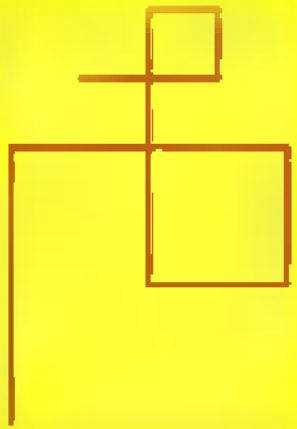
squaggle



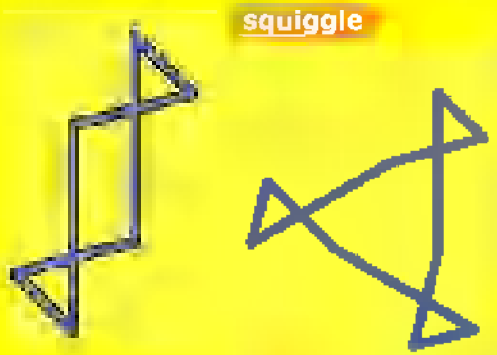
squoggle



design

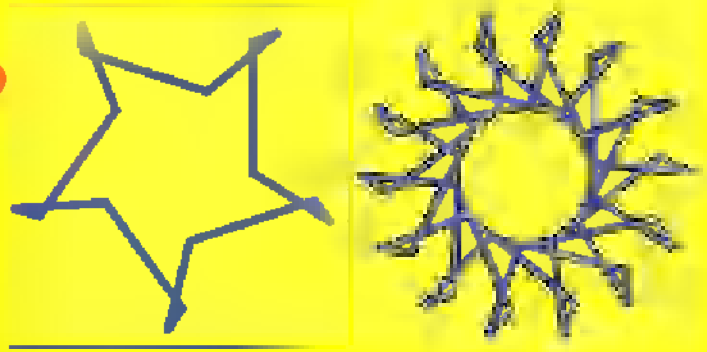


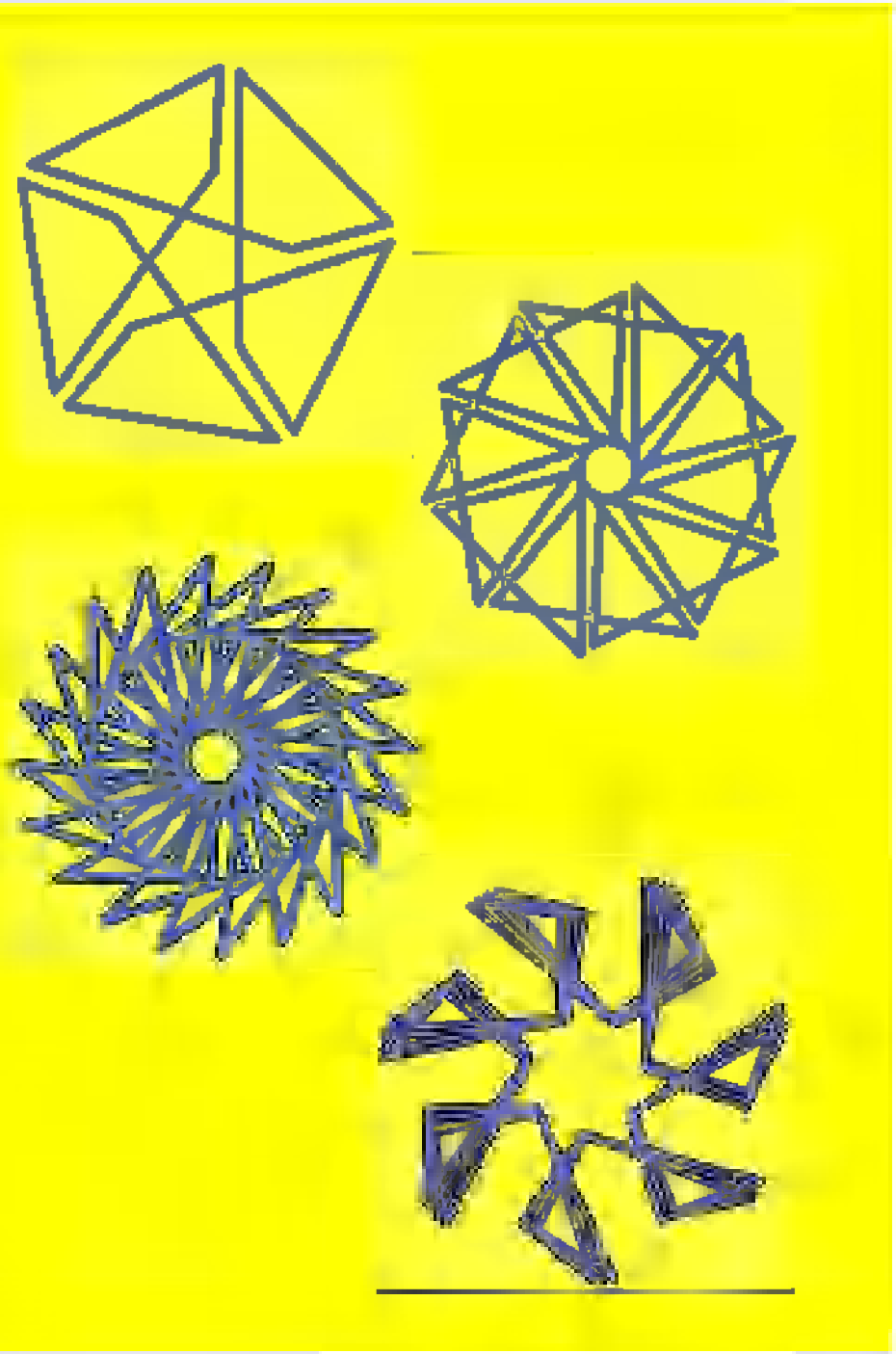
squiggle

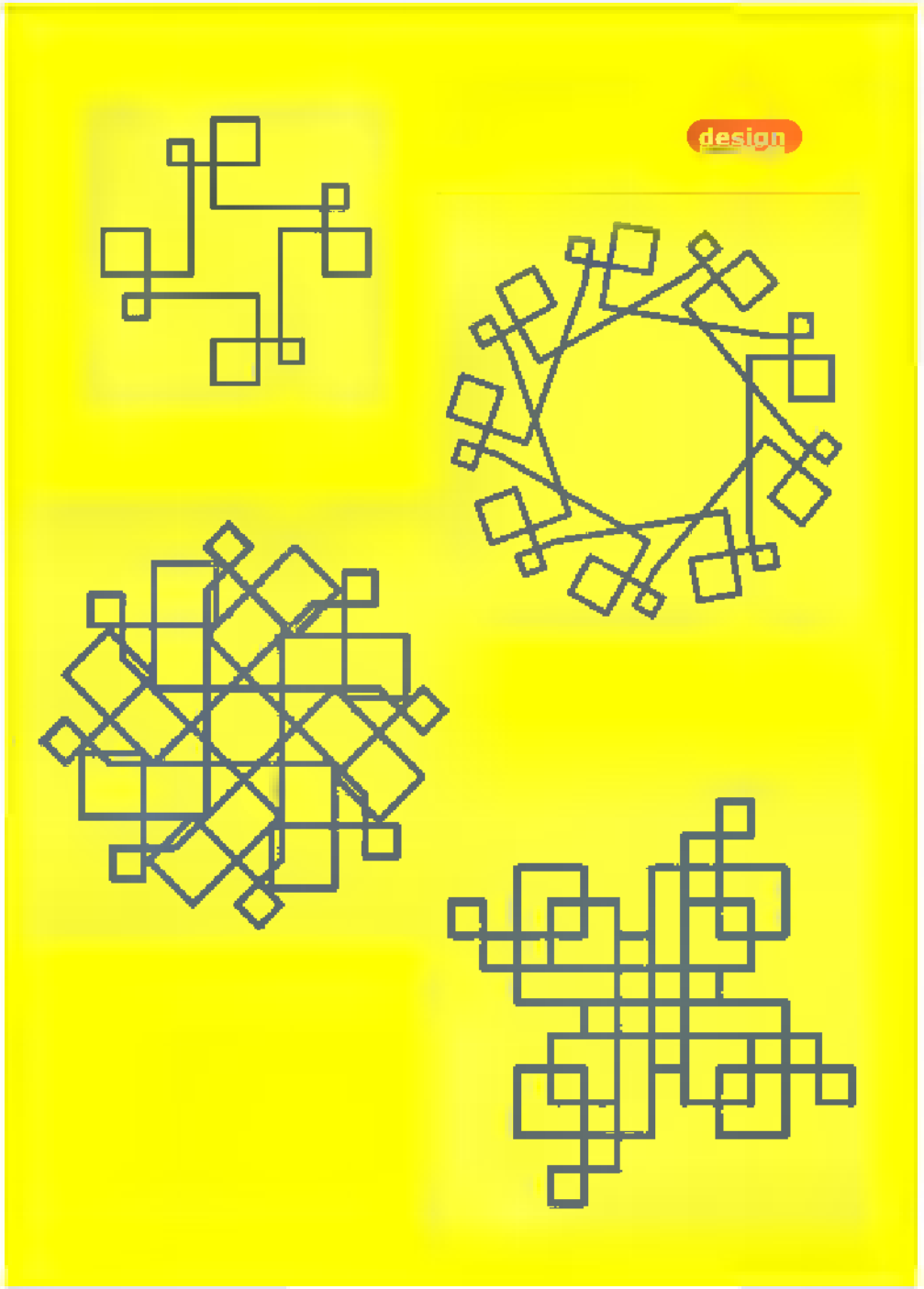


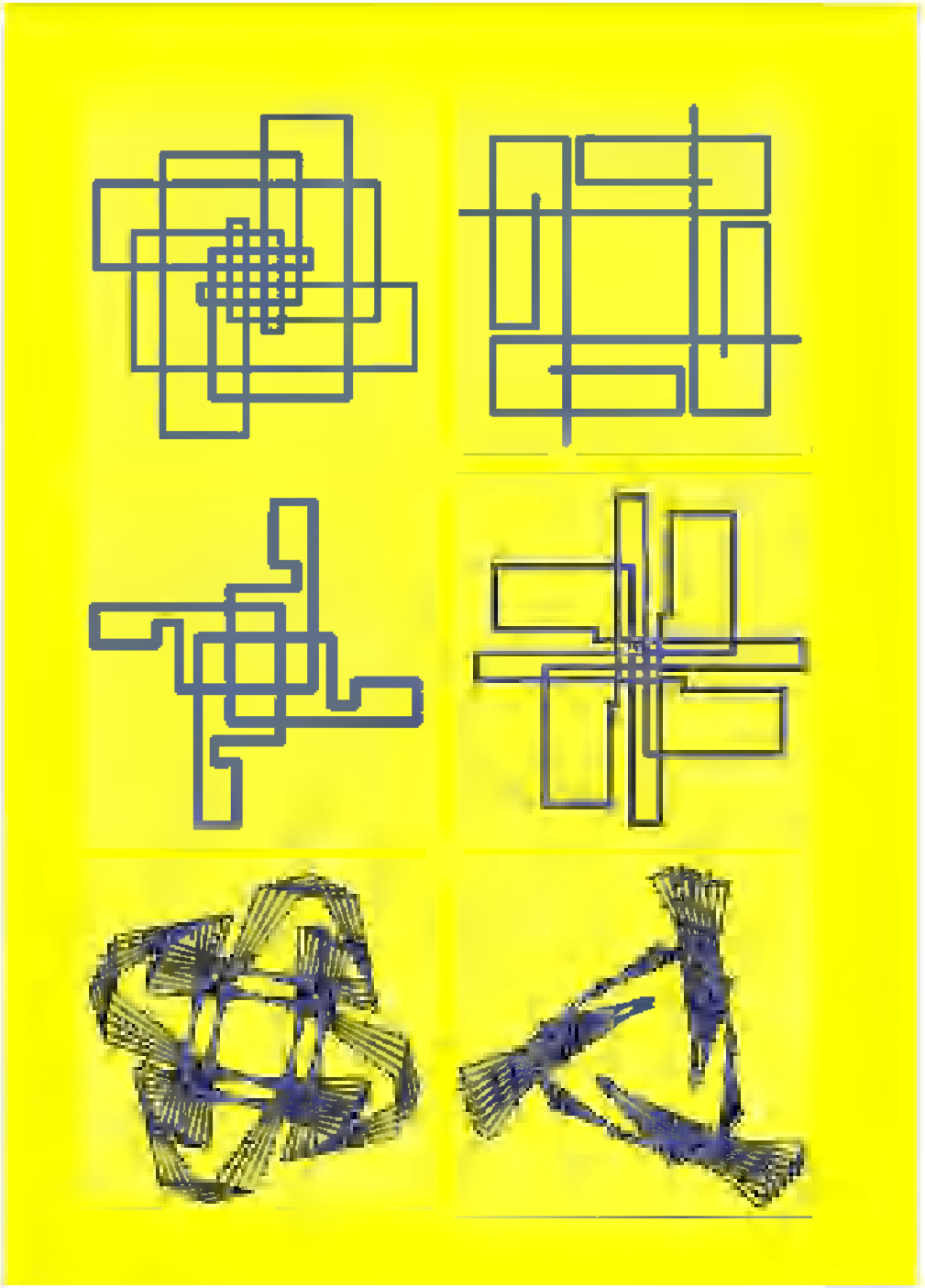
squaggle

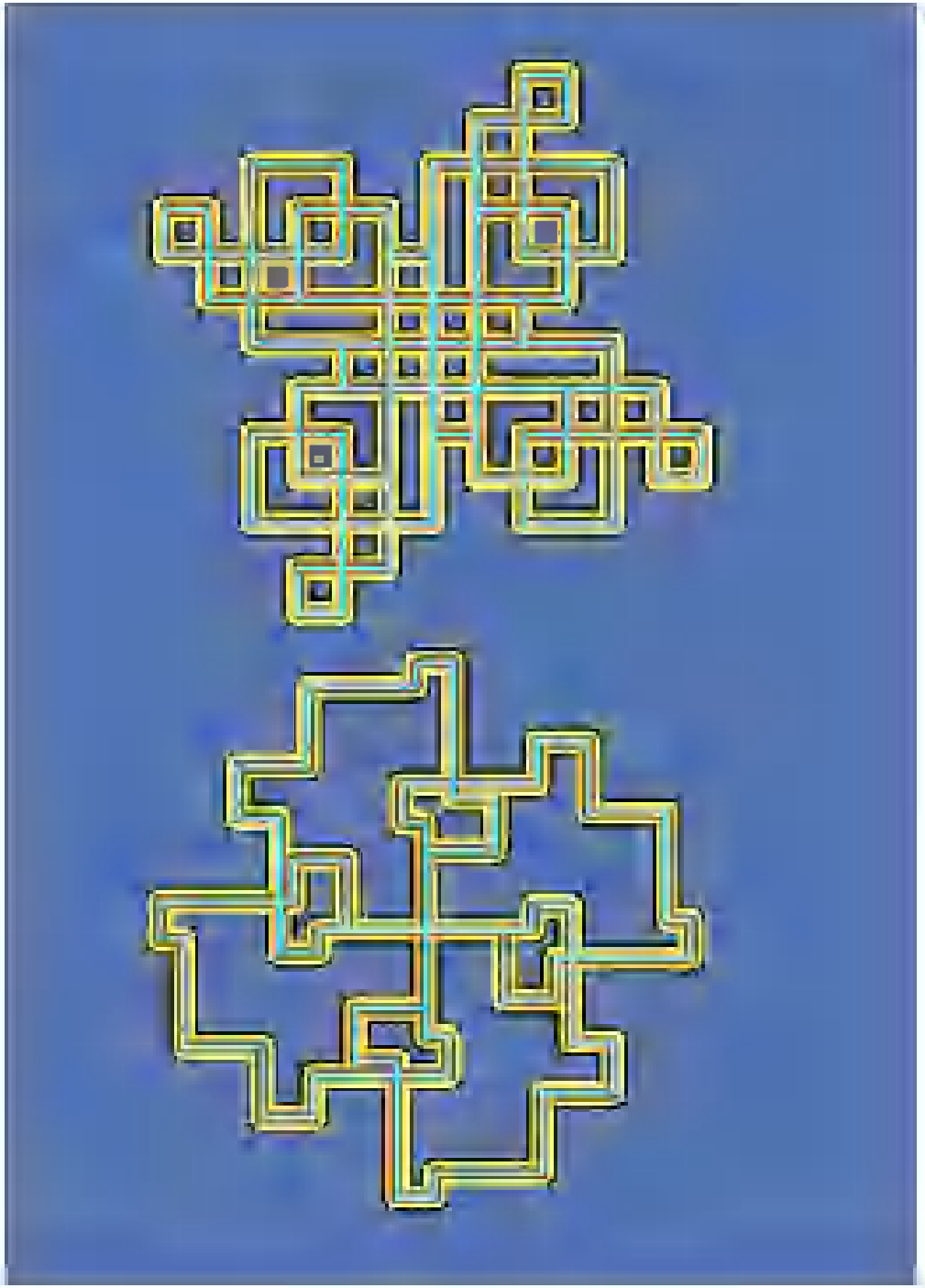
squoggle

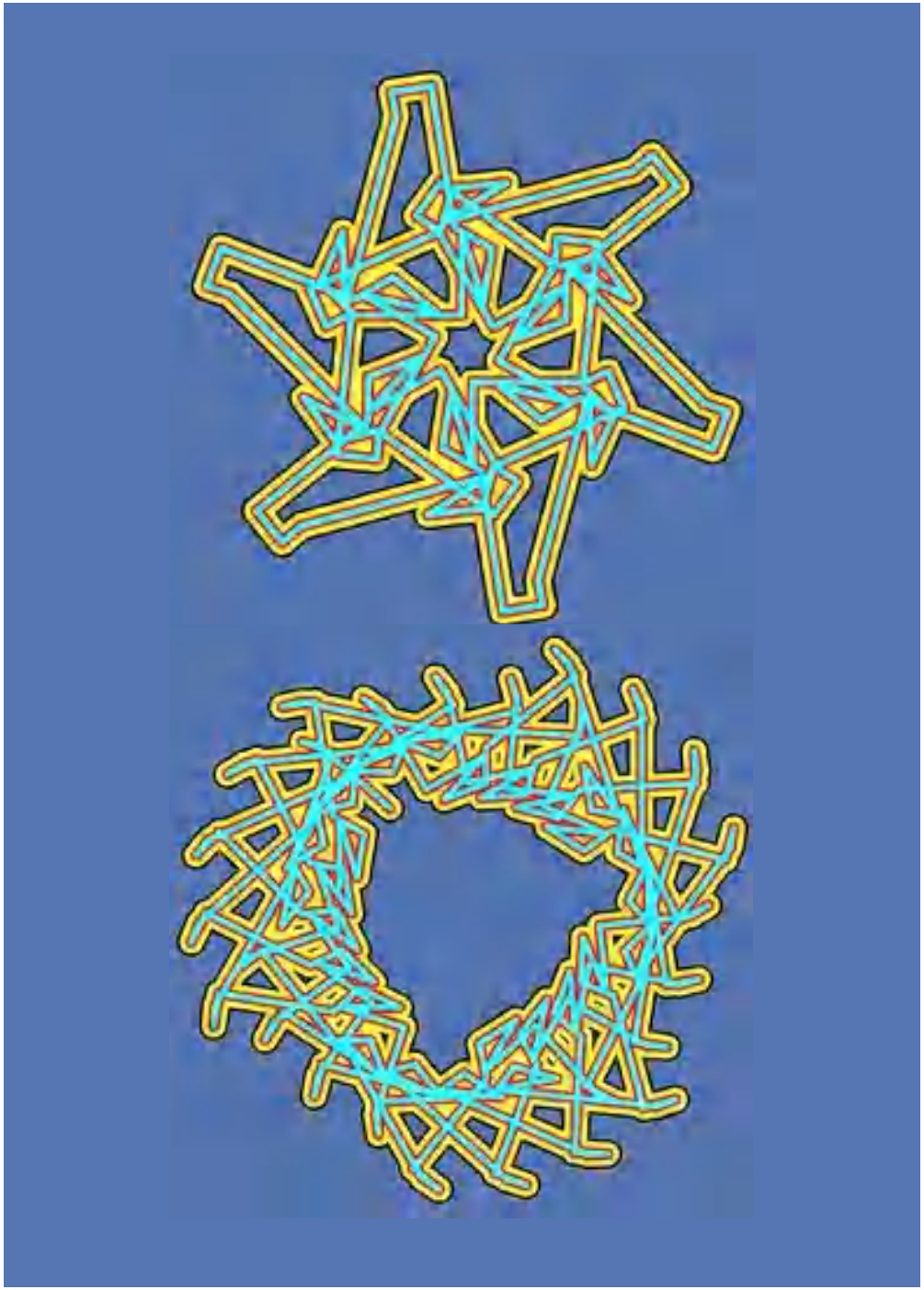





design







Kreise (I)

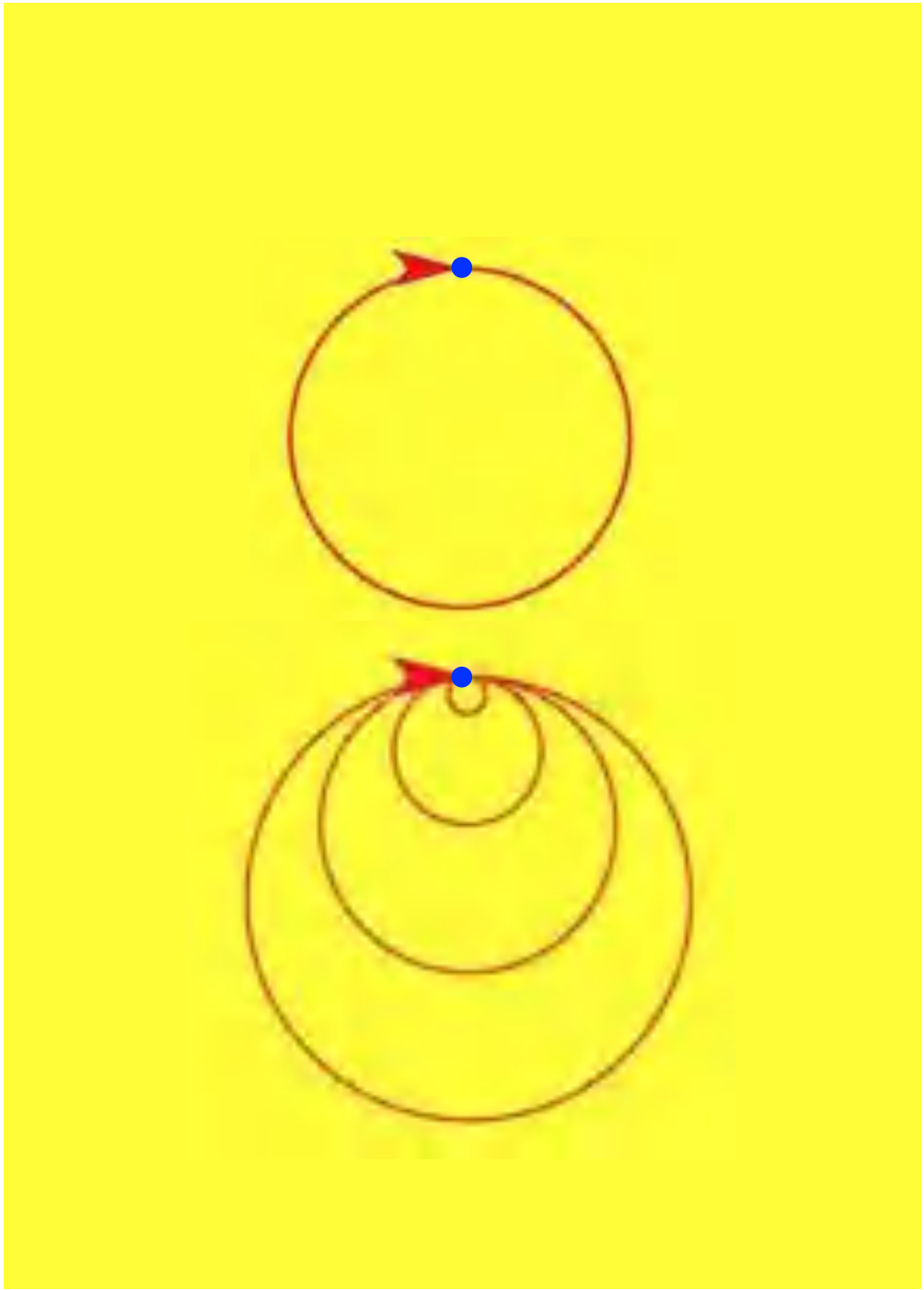
Beim Zeichnen von Figuren ließ Papert gerne die Kinder selbst Schildkröte spielen. Er spricht von **körper-syntonischem Lernen** bzw. **ego-syntonischem Lernen**, um die Beziehung zwischen den geometrischen Operationen und der körperlichen Erfahrung deutlich zu machen.

Ein Paradebeispiel dafür ist das Zeichnen eines Kreises: Aus der wiederholten Abfolge der Körperbewegungen *„mache kleinen Schritt nach vorne, drehe dich ein wenig“* ensteht tatsächlich ein Kreis.

Von der verbalen Beschreibung ist es nur noch ein kleiner Schritt zum Programm. Eine wiederhole-Schleife mit dem **wiederhole**-Block zeigt das erwartete Ergebnis.

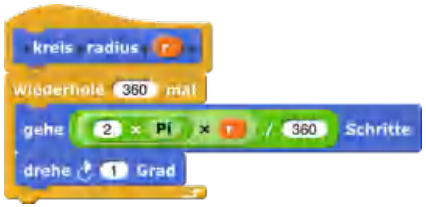
Sollen unterschiedlich große Kreise gezeichnet werden, kann dies auf ganz unterschiedliche Weise erreicht werden. Eine Möglichkeit ist eine Prozedur **kreis**, bei der über die Zahl der **schritte** der Schildkröte der Radius des Kreises bestimmt wird:

Die Prozedur **kreis** ähnelt damit stark der Prozedur **polygon**, wobei hier der Winkel konstant bei einem Grad gehalten wird.



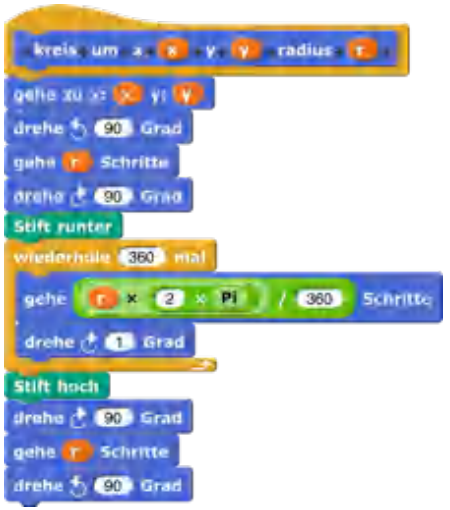
Kreise (II)

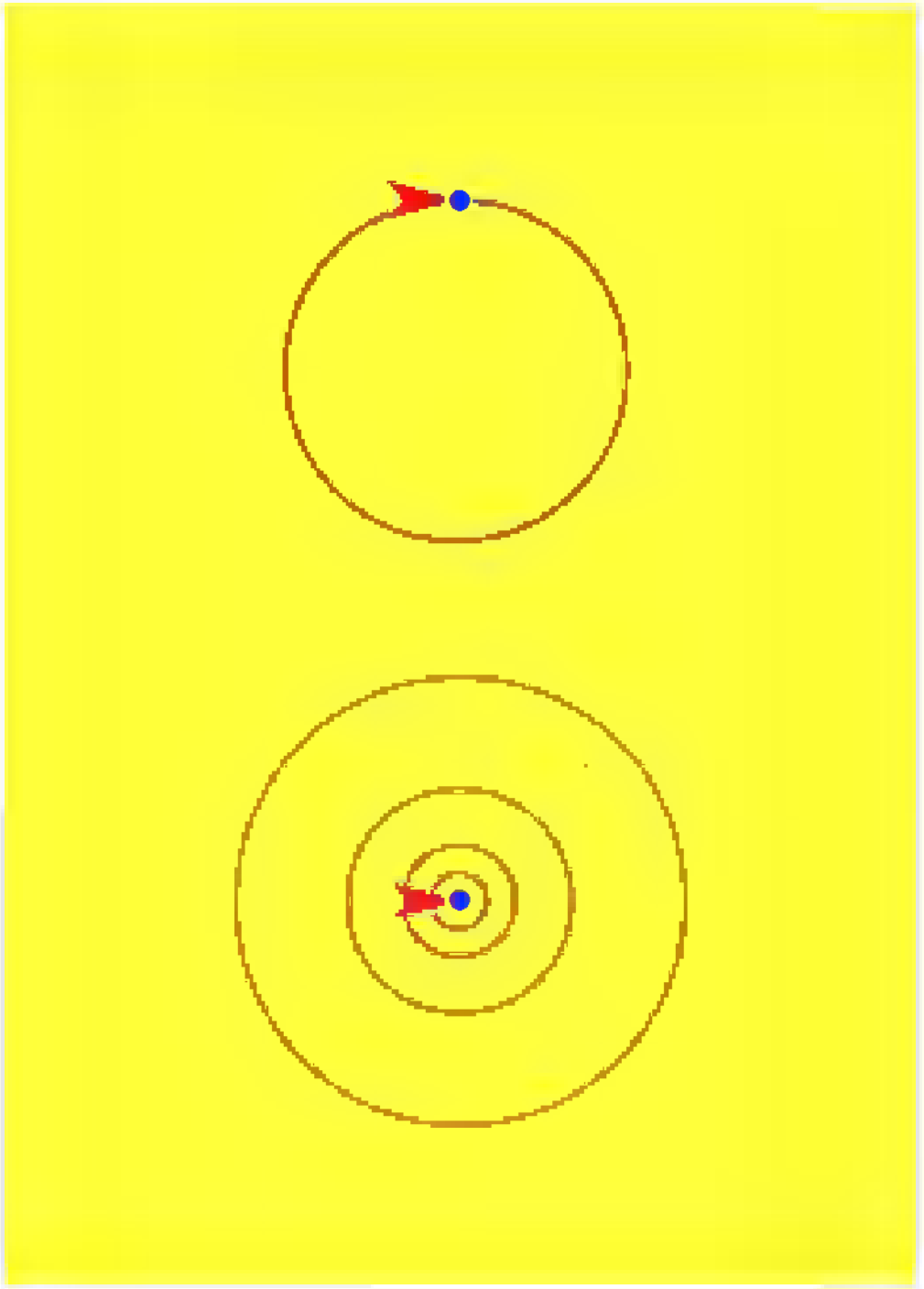
Mit ein wenig Mathematik können die Kreise genauer bestimmt werden. Aus der Formel für den **Kreisumfang** und der **Kreiszahl π** lassen sich über den **Radius** die erforderlichen Schrittgrößen der Schildkröte berechnen: **U = 2 π r**. Das ergibt die Prozedur **kreis radius**:

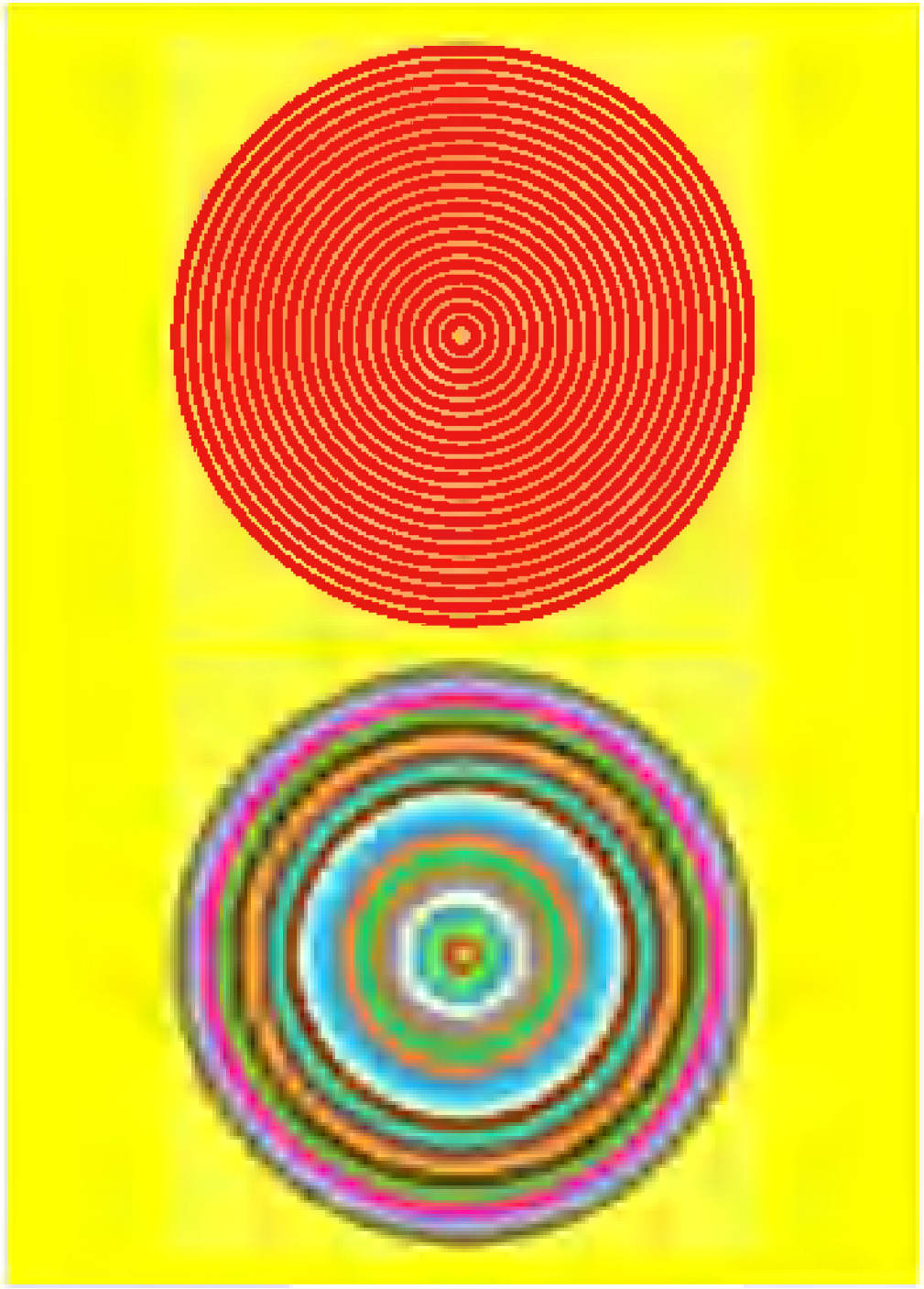


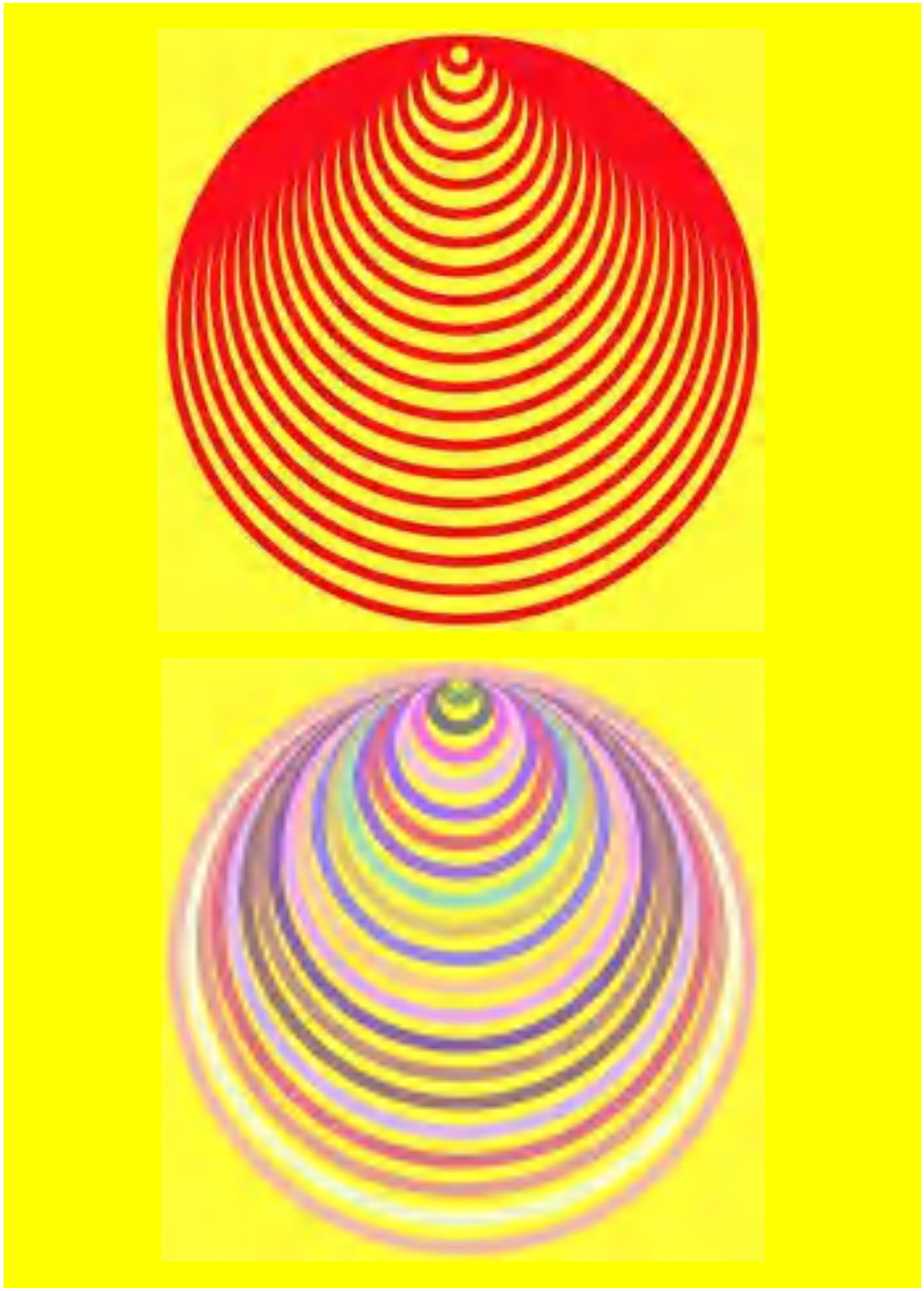
Beim Zeichnen von Kreisen auf Papier wird ihre Lage mit dem Lineal und ihr Radius mit dem Zirkel um den gewählten Mittelpunkt bestimmt. Eine Prozedur **kreis um ...**, die diesem Vorgehen entspricht, ist etwas umfangreicher:

- Zunächst wird die Schildkröte zum gewählten Punkt **[x,y]** geschickt.
- Von diesem Ausgangspunkt wird sie mit **radius** zum Startpunkt auf dem Kreis geschickt und
- zeichnet von dort den Kreis.
- Abschließend wird die Schildkröte zum Ausgangspunkt zurück geschickt.

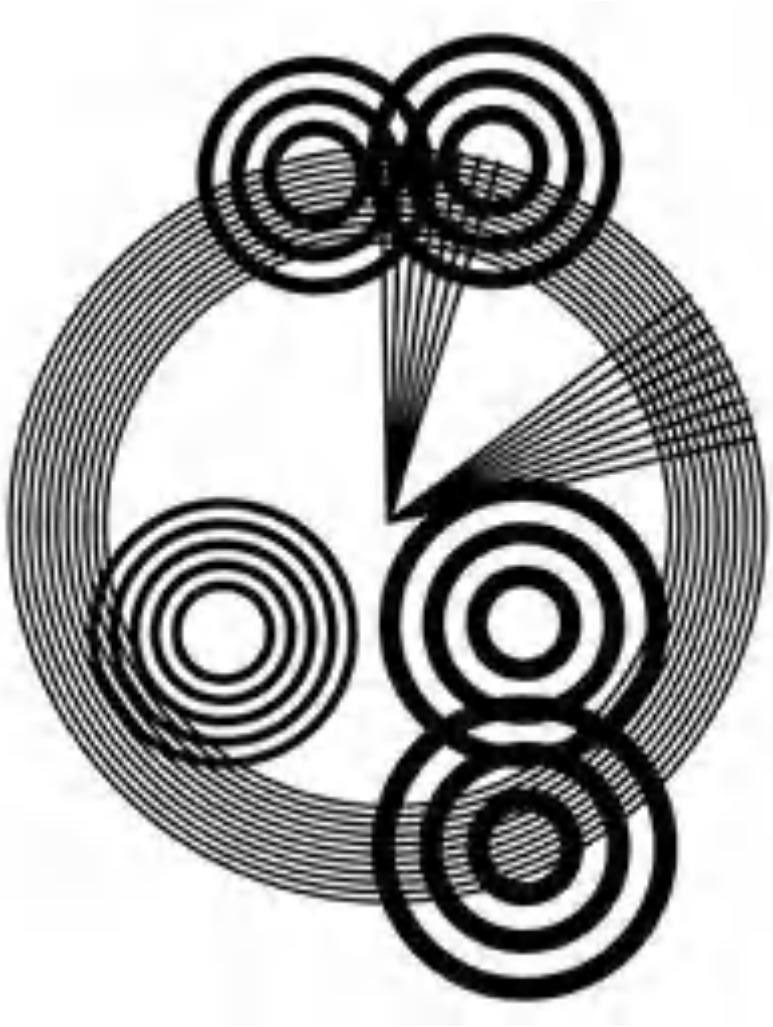








Linien und Kreise



Hommage à Delaunay: Rhythmus

Hommage à Calder

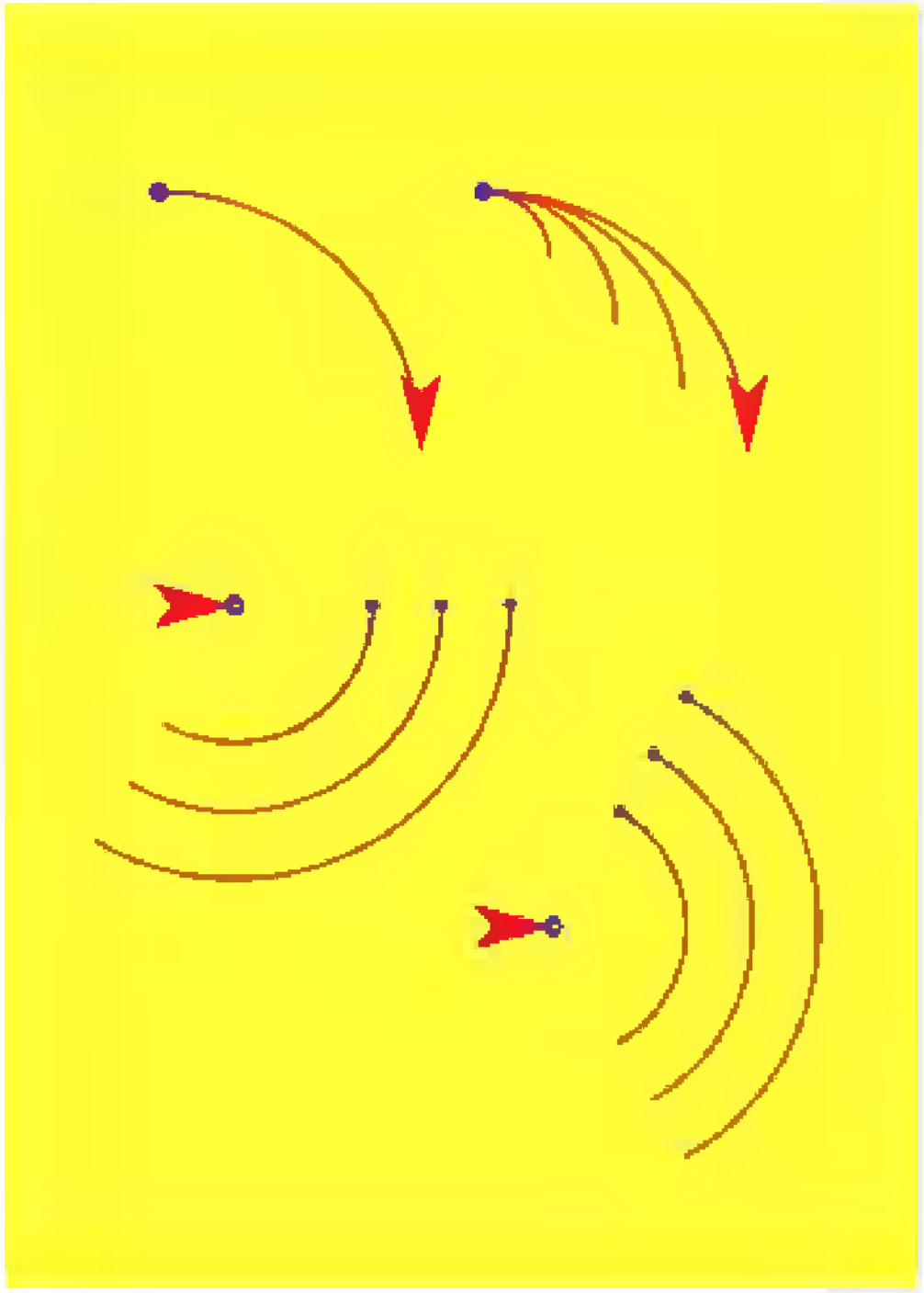
Kreisbögen

Kreise sind das Paradebeispiel für die von Papert beschriebene Beziehung zwischen geometrischen Operationen und der körperlichen Erfahrung. An ihnen kann auch gezeigt werden, dass durch **Weglassen** neue **Vielfalt** entstehen kann. So entstehen aus Kreisen durch eine Beschränkung der Schritte und Drehungen Kreisbögen.

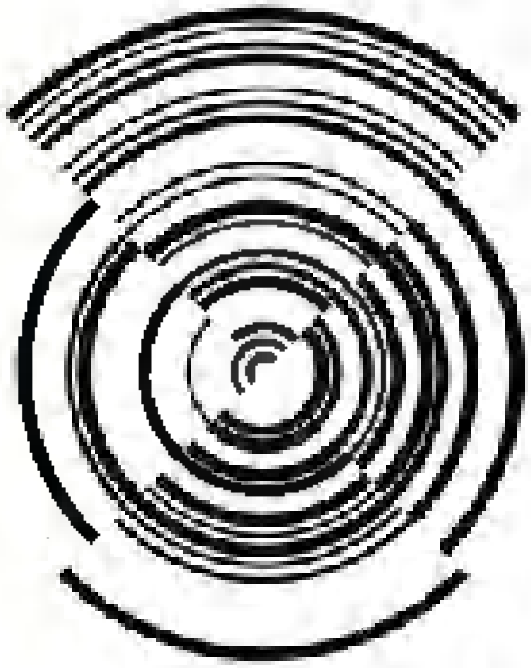
Wie bei den Kreisen gibt es wieder mehrere Möglichkeiten, sie von einem Punkt ausgehend zu erzeugen.

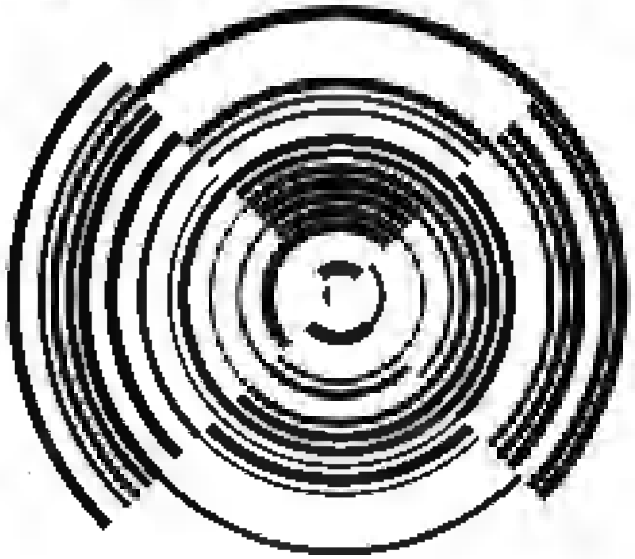
Auch bei Kreisbögen ist das Zeichnen mit einem gewählten Mittelpunkt naheliegend. Dabei sind Bögen, die **von** einem Punkt ausgehen, zu unterscheiden von Bögen, die **um** einen Punkt gehen. Diese Prozeduren sind nun etwas umfangreicher:

Durch ihre Kombination entstehen viele neue interessante Figuren.



Kreisbögen 90°





Kreisbögen 30° < zufällig < 60°

Kreisbögen 180°

verschobene Kreisbögen 180°

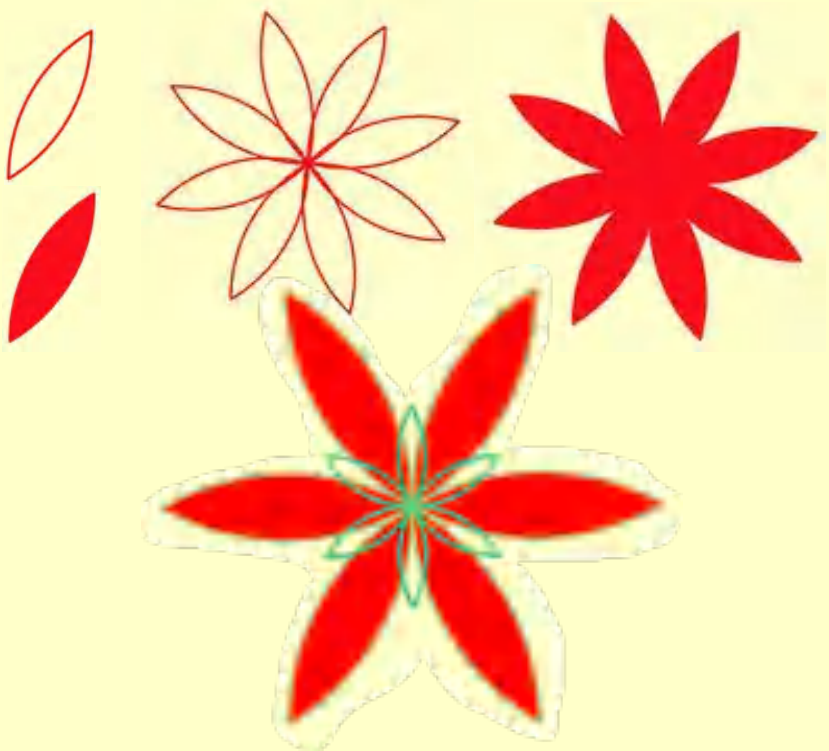
Kreisbögen - Anwendung mit Variationen

Kreisbögen können als Grundlage für komplexere Figuren genommen werden. Sehr einfach lassen sich so Blätter zeichnen, die wiederum zu Blumen kombiniert werden können.

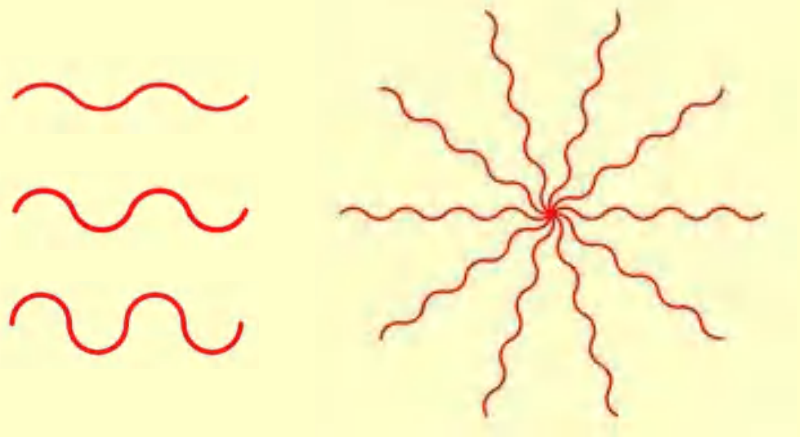
Mit dem Befehl **male aus** können auch farbig gefüllte Blätter erstellt und in entsprechenden Kombinationen eingesetzt werden.

Weitere Varianten werden möglich, wenn zwischen rechts- und linksdrehenden Kreisbögen unterschieden wird (die sich intern aber nur durch die Befehle **drehe 1 Grad** beziehungsweise **drehe -1 Grad** unterscheiden). Daraus können ganz verschiedene Wellen entstehen.

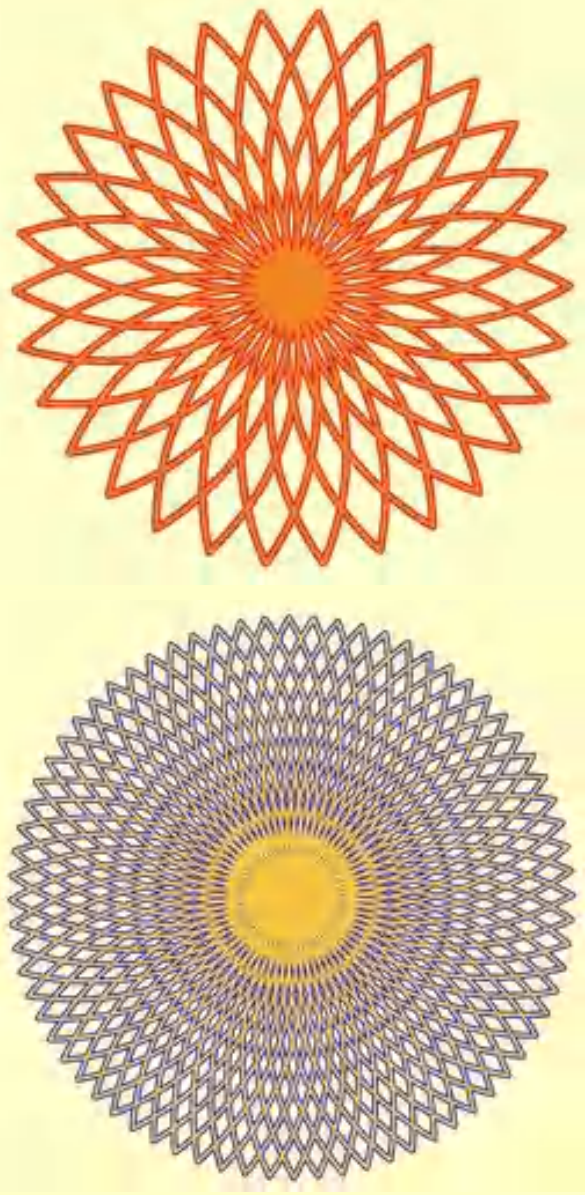
Blatt & Blüte (Kreisbögen 120°)



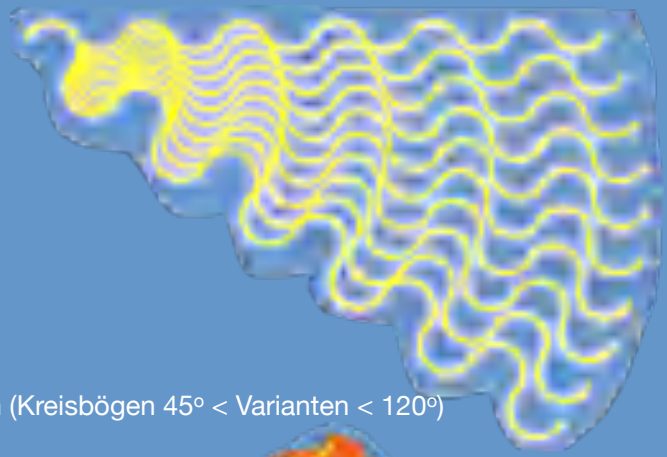
Wellen (Kreisbögen 45° < Varianten < 180°)



Blüten (Kreisbögen 120°)

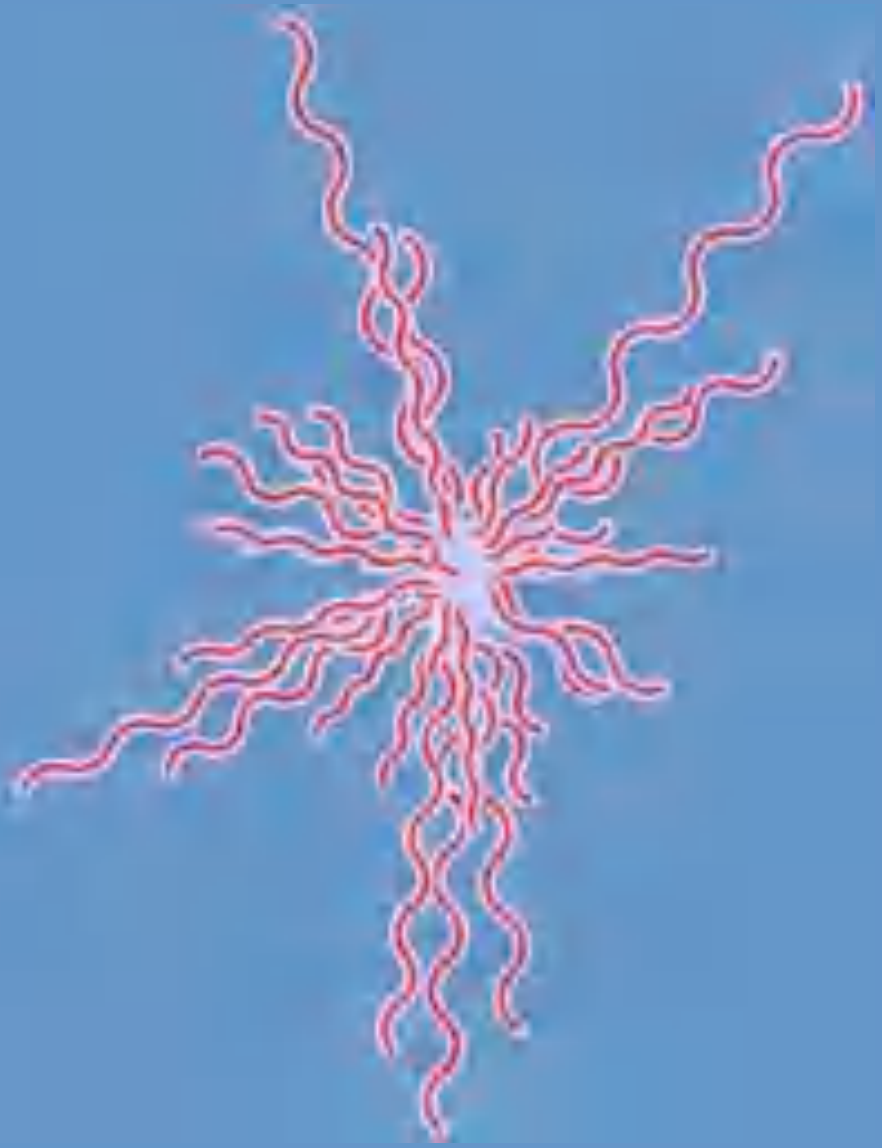


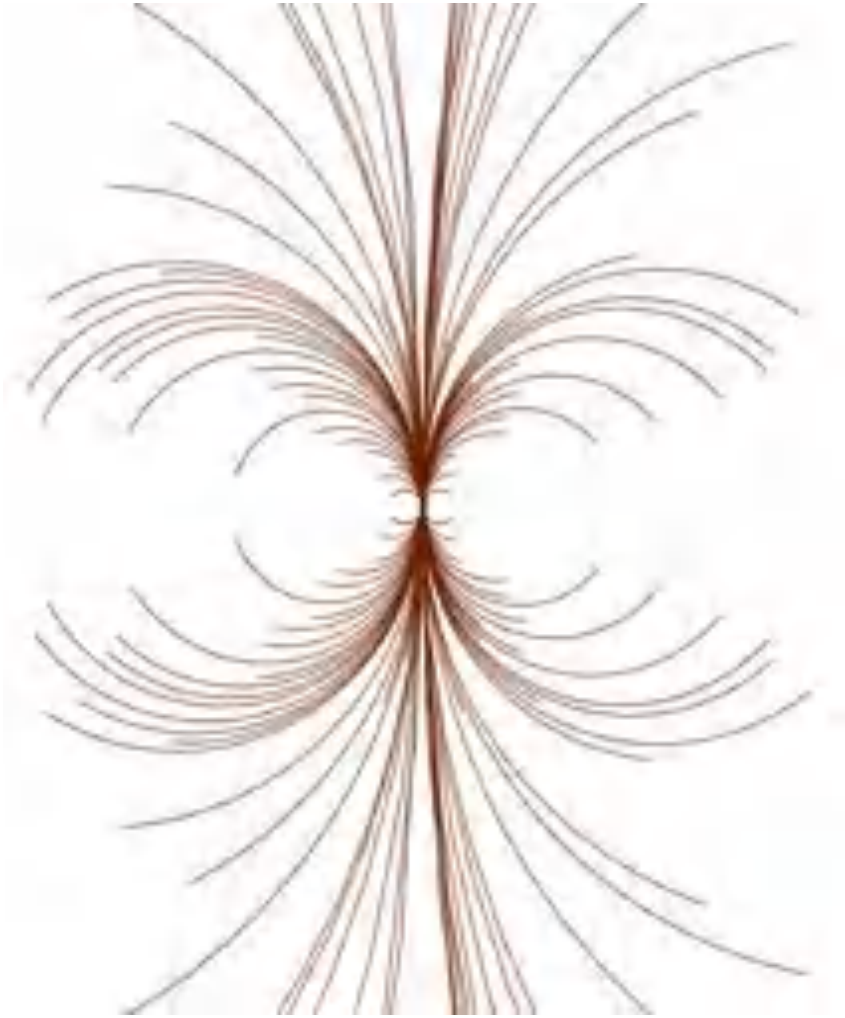
Wellen (Kreisbögen 120° < Varianten < 200°)



Vulkan (Kreisbögen 45° < Varianten < 120°)

Amöbe (Kreisbögen 45° < Varianten < 120°)





Hommage à Nees: Kreisbögen (Kreisbögen 90° - 360°)

Hommage à Sýkora: Linien (Kreisbögen 30° - 120°)

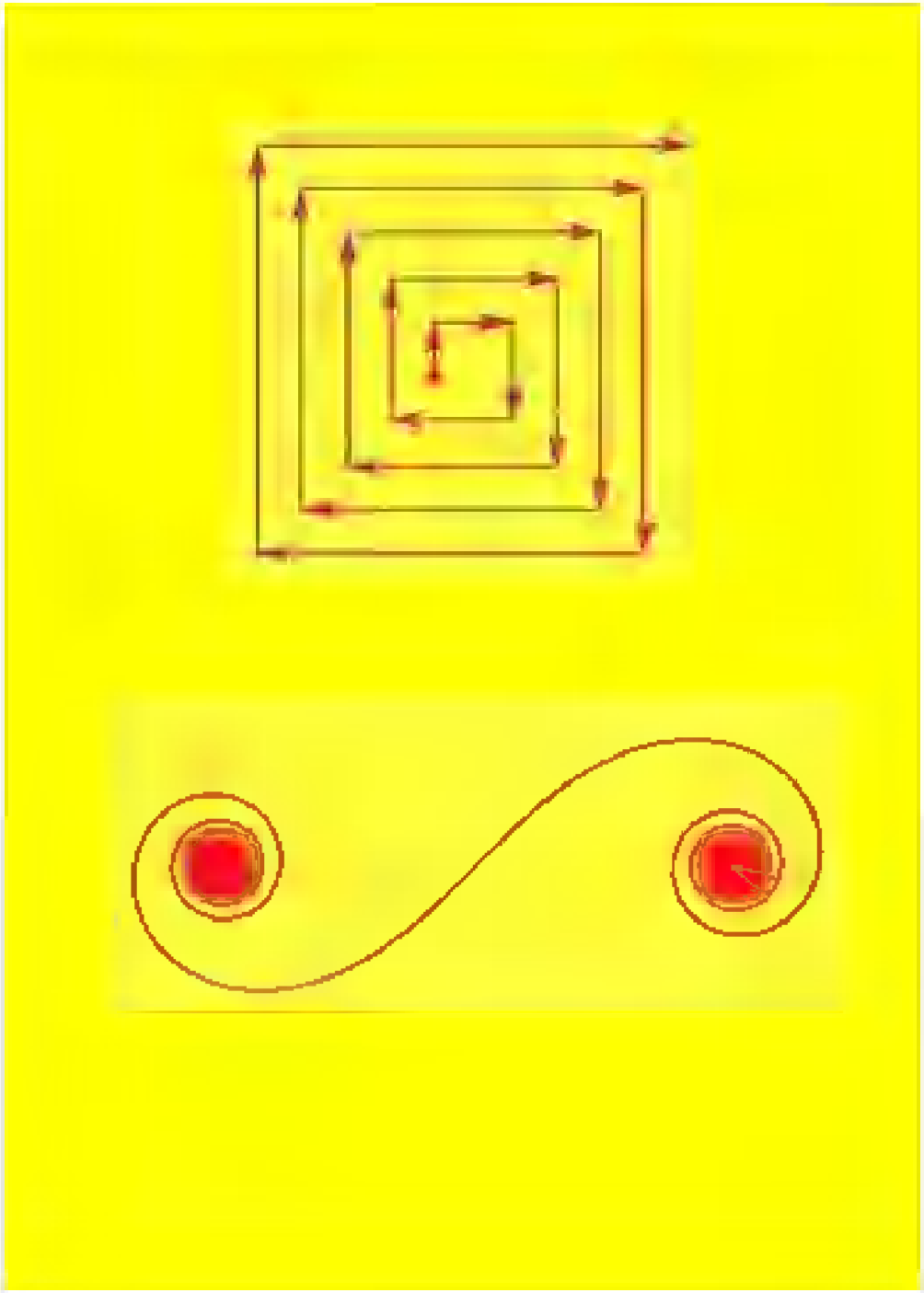
Spiralen

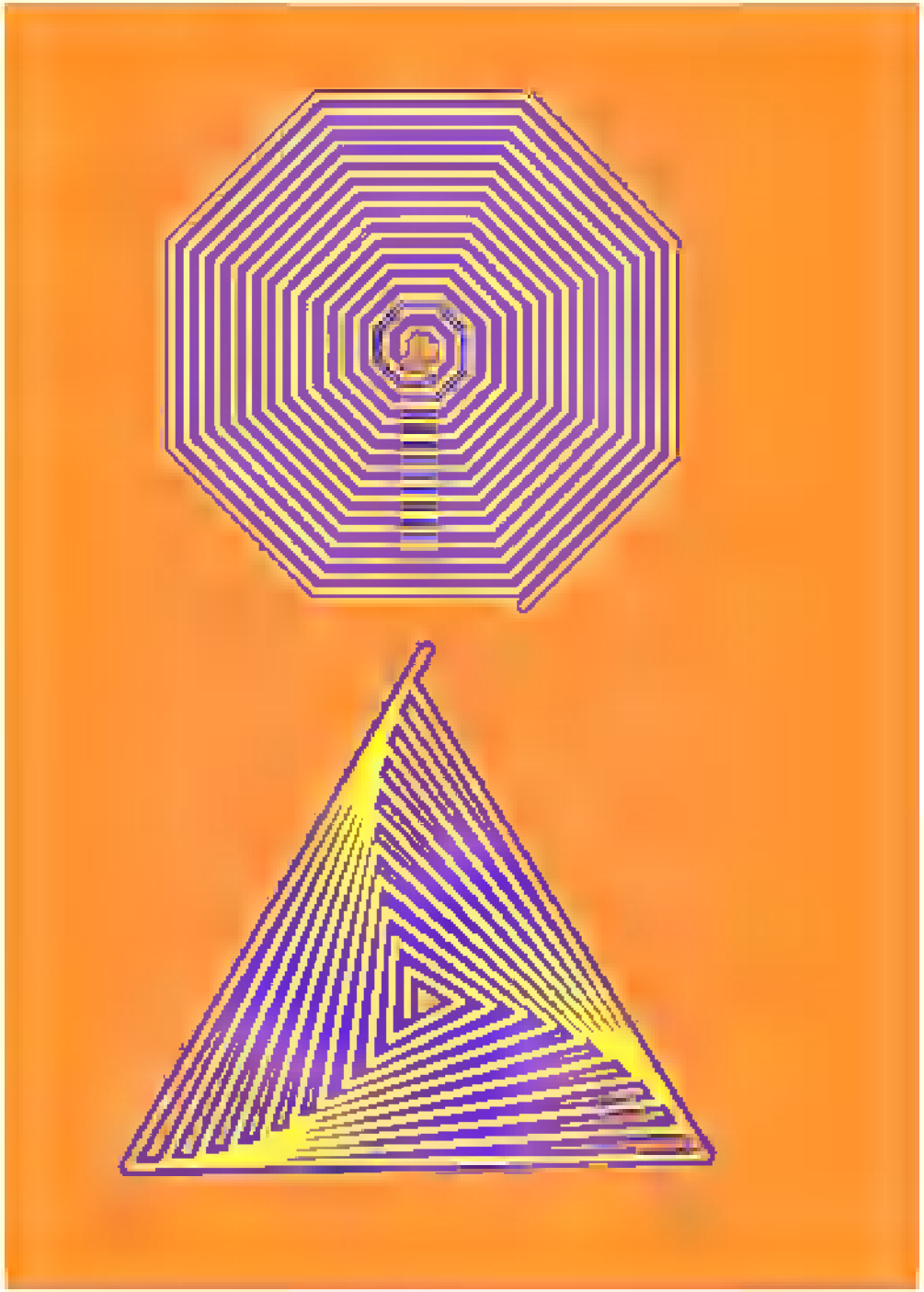
Wenn in den regelmäßigen Vielecken (Polygonen) die **Längen** der Seiten und die **Winkel** zwischen den Seiten **veränderbar** werden, erhalten wir **Spiralen**. Dafür gibt es mehrere Möglichkeiten, die interessante Unterschiede ergeben.

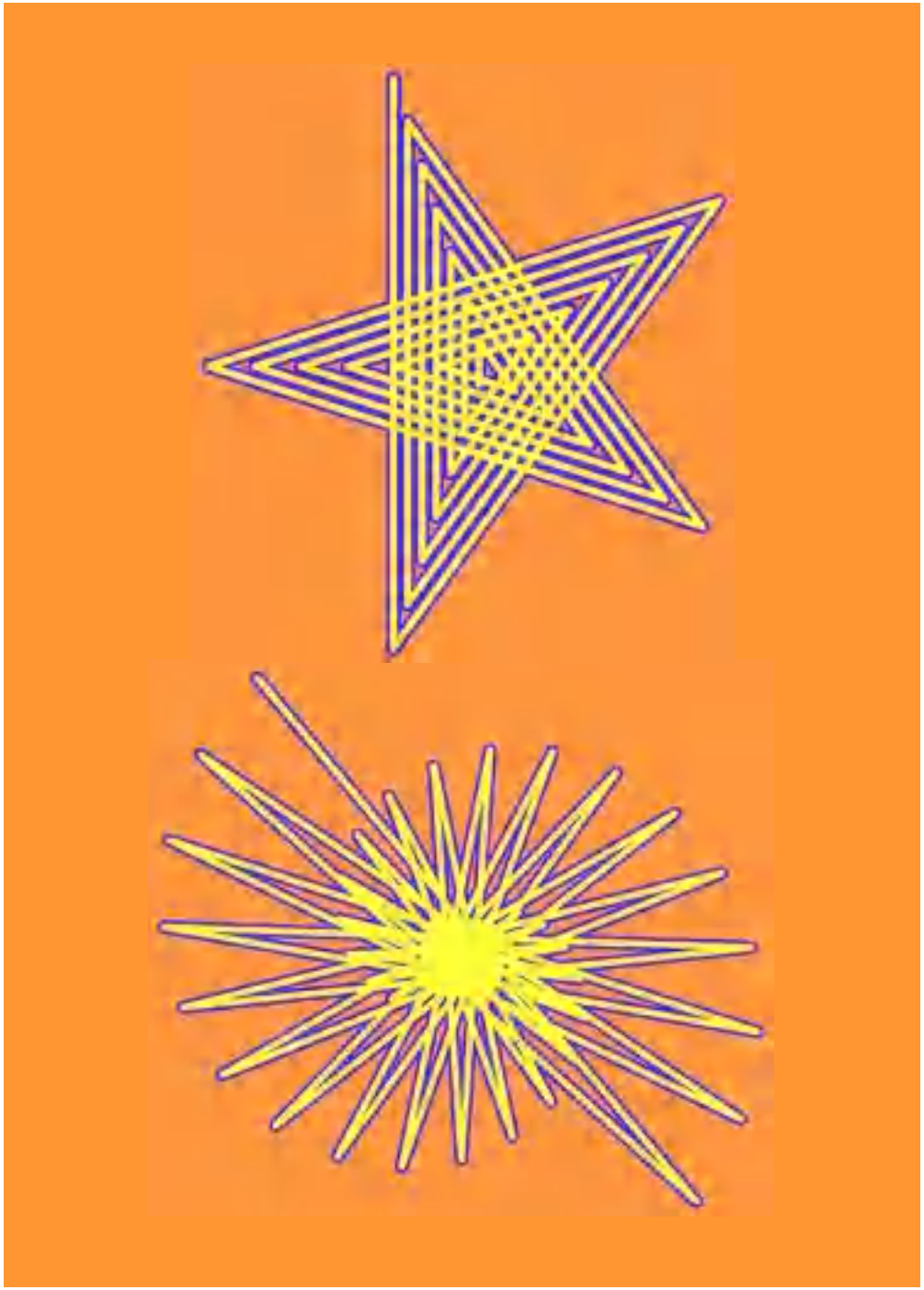
Ähnlich wie in der Prozedur **polygon** geht die Schildkröte eine Strecke und dreht sich nach jedem Schritt um einen Winkel. Allerdings wird in der neuen Prozedur **spirale** die Strecke jeweils um einen Faktor **delta d** erhöht (Bild rechts oben). Damit diese Prozedur nicht endlos weiter läuft, ist eine **grenze g** für die Strecke vorzugeben. Wenn die **Bedingung l > g** also den Wert **wahr** liefert, ist diese **grenze** überschritten und die Prozedur wird abgebrochen.

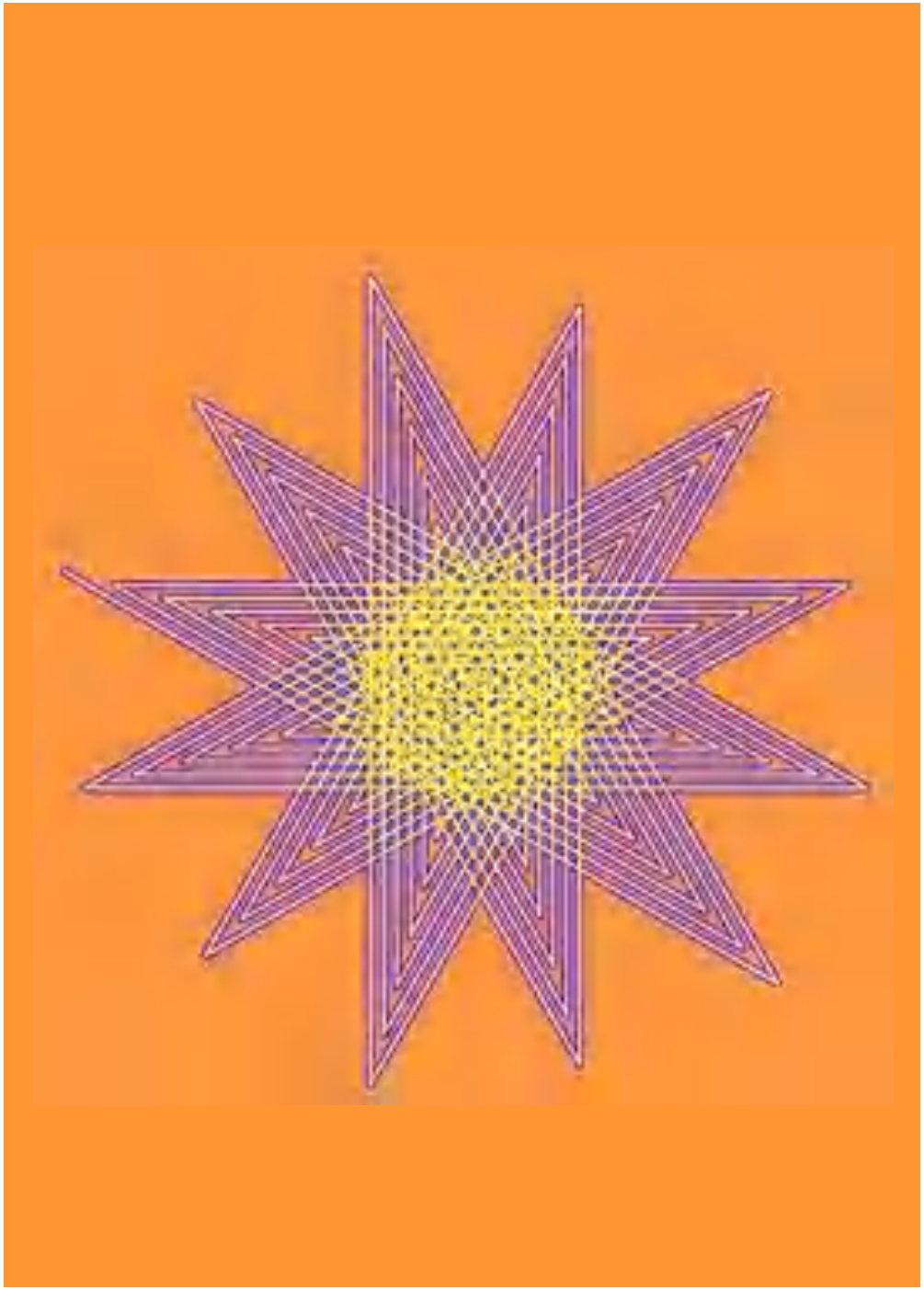
Statt der Strecke kann natürlich auch der Drehwinkel erhöht werden. Dazu reicht die Änderung des Befehls **ändere l um d** in **ändere w um d**.

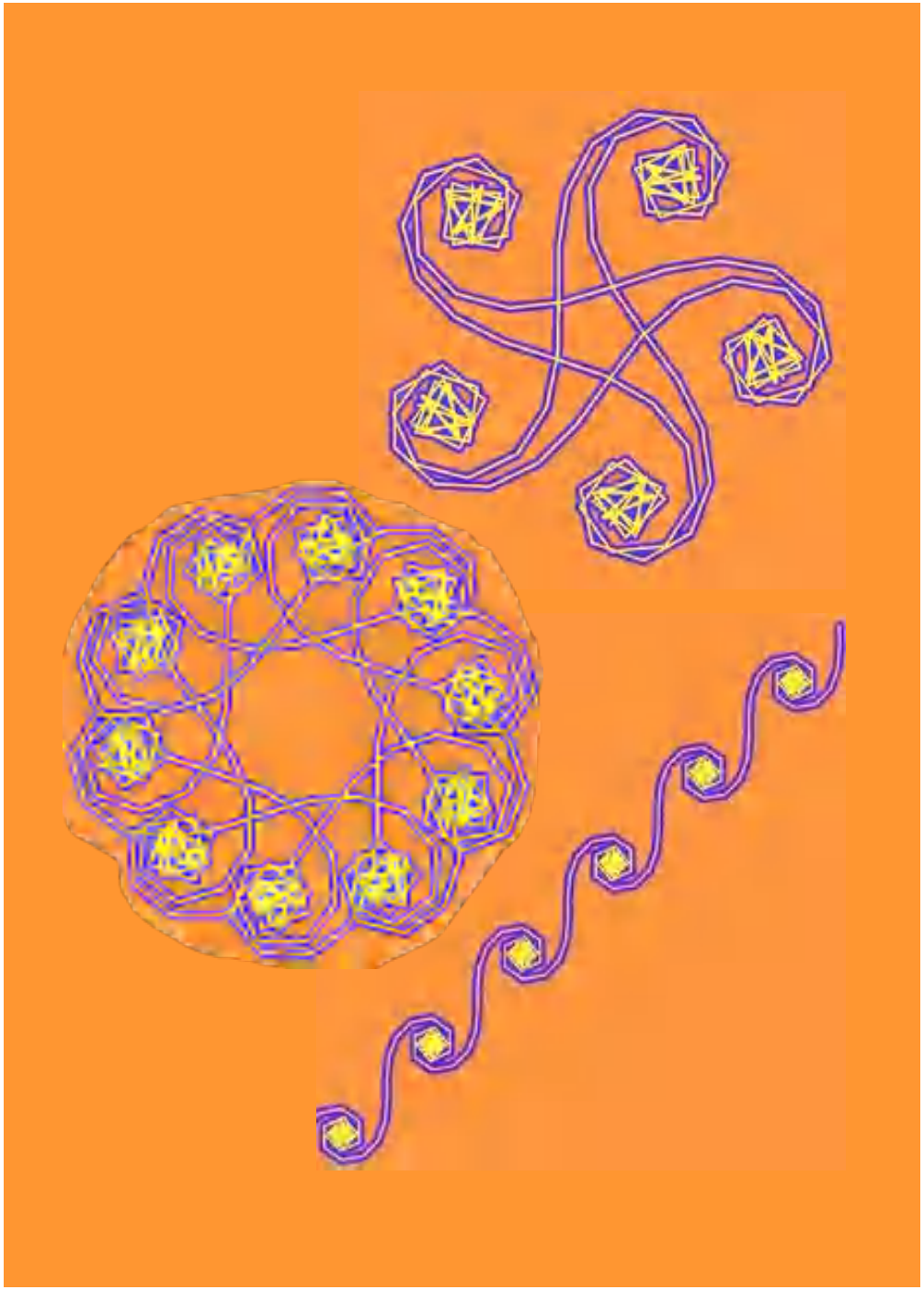
Das Ergebnis ist dann ein völlig anderes (Bild rechts unten). Es handelt sich bei der Kurve um eine Klothoide (auch als Cornu-Spirale oder Euler-Spirale bekannt), bei der die Krümmung linear mit der Weglänge wächst.











Rekursive Spiralen

An Spiralen wird gerne die Rekursion als eine Form der **Wiederholung** eingeführt. Rekursive Prozeduren können einen oder mehrere **Selbstaufrufe** enthalten. Das Prinzip ist besonders leistungsfähig, weil bei diesem Aufruf (auch veränderte) Parameter übergeben werden können.

Damit die Aufrufe nicht endlos weiter gehen, ist eine **grenze g** vorzugeben. Liefert die entsprechende Bedingung den Wert **wahr**, ist diese **grenze** überschritten und die Aufrufe werden beendet. Je nach Ort des Selbstaufrufs innerhalb der Prozedur werden drei Arten der Rekursion unterschieden:

anfangsständige Rekursion:

- Abbruchbedingung
- **Selbstaufruf** der Prozedur
- Anweisung(en)

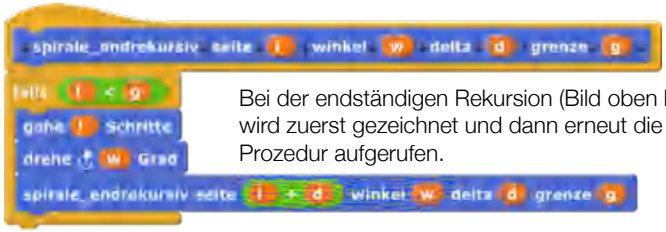
mittelständige Rekursion

- Abbruchbedingung
- Anweisung(en)
- **Selbstaufruf** der Prozedur
- Anweisung(en)

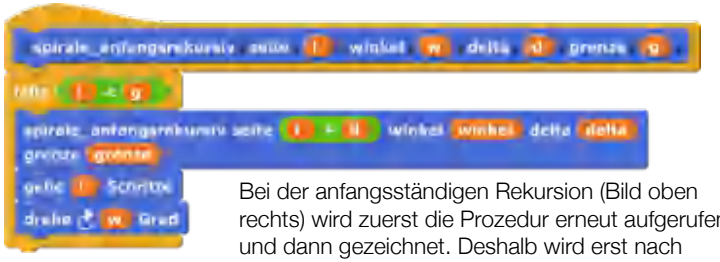
endständige Rekursion:

- Abbruchbedingung
- Anweisung(en)
- **Selbstaufruf** der Prozedur

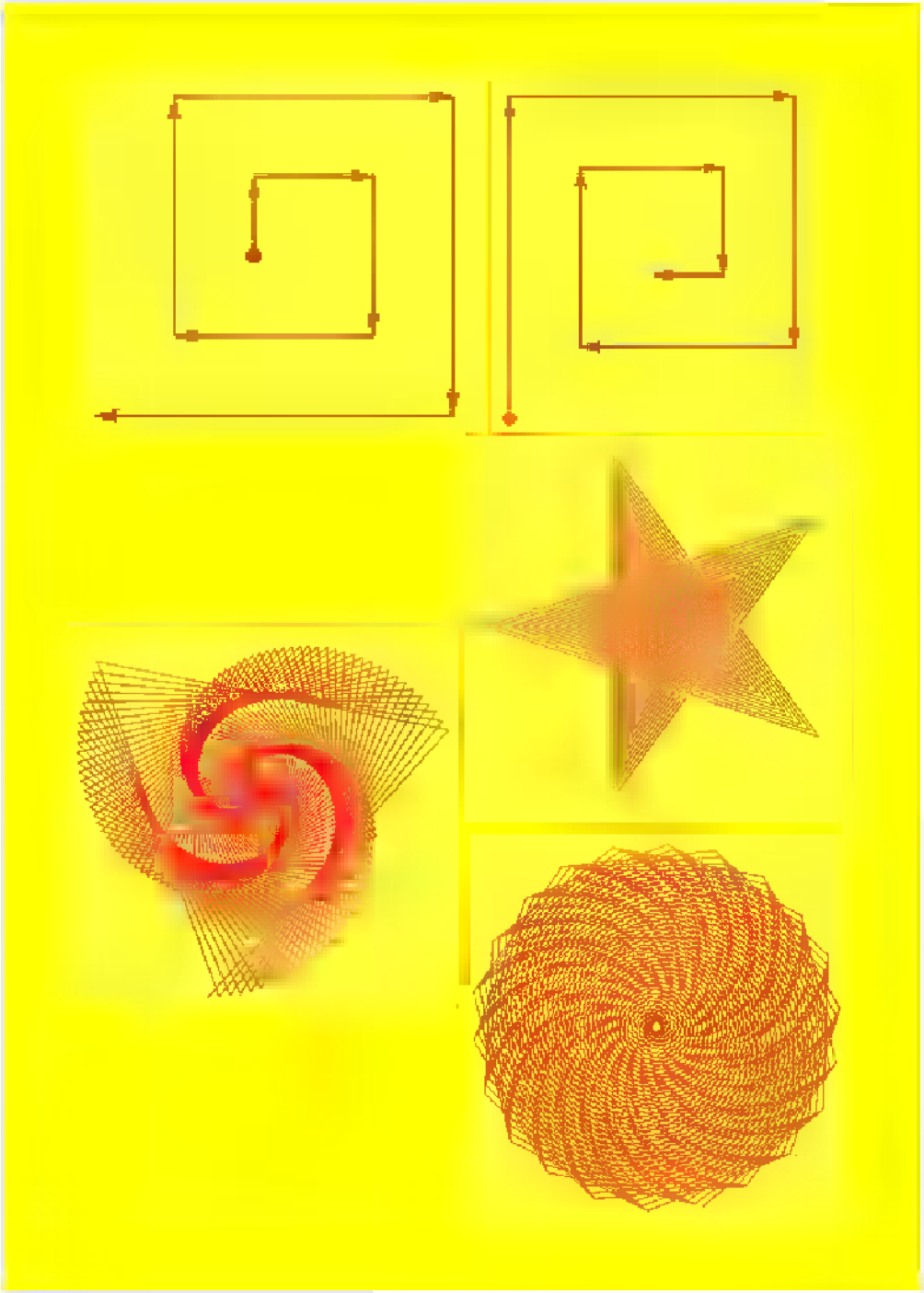
An den Spiralen ist der Ablauf gut nachzuvollziehen:

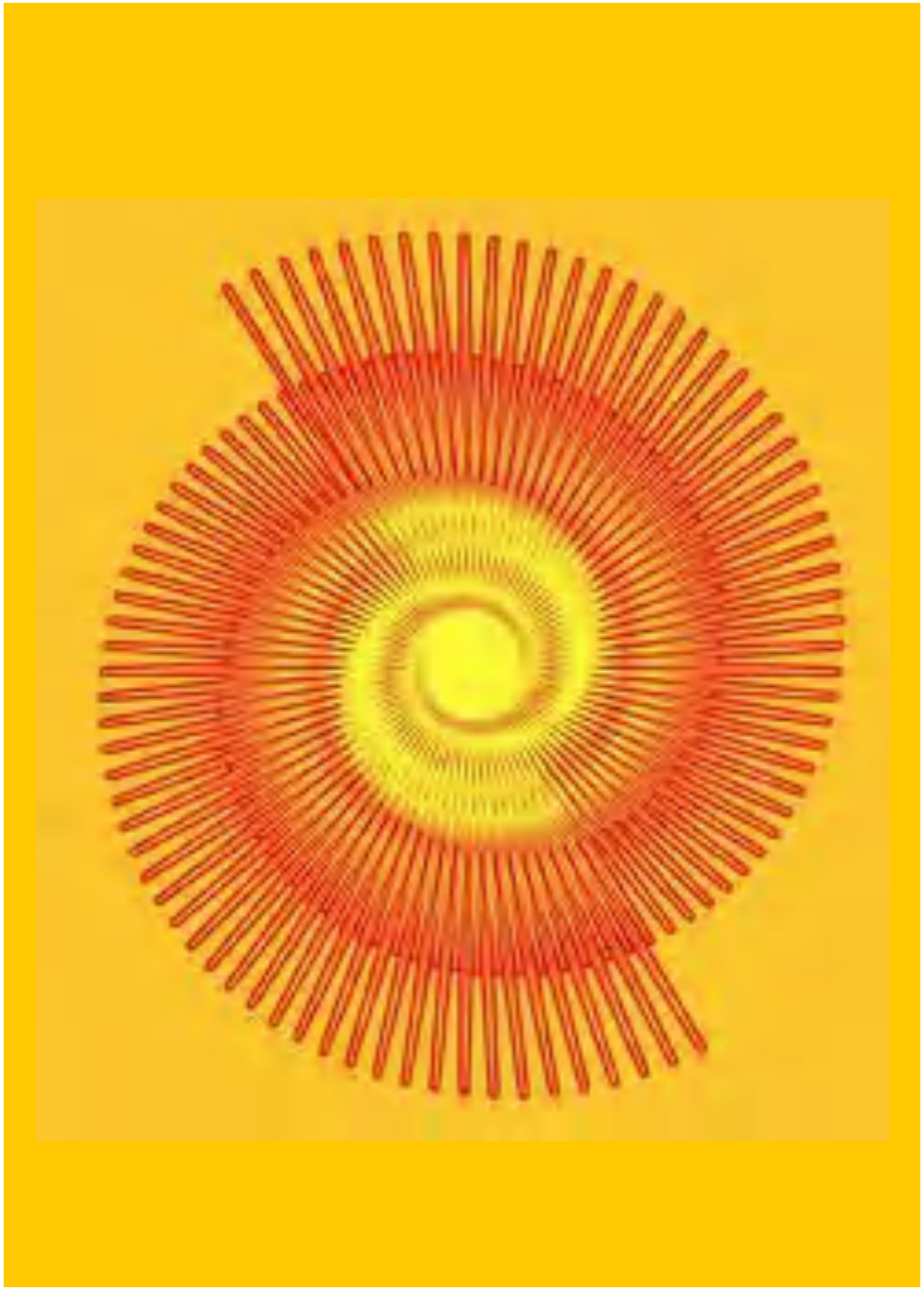


Bei der endständigen Rekursion (Bild oben links) wird zuerst gezeichnet und dann erneut die Prozedur aufgerufen.



Bei der anfangsständigen Rekursion (Bild oben rechts) wird zuerst die Prozedur erneut aufgerufen und dann gezeichnet. Deshalb wird erst nach erfülltem Abbruchkriterium im „Rücklauf“ mit der bis dort aufsummierten Länge gezeichnet.



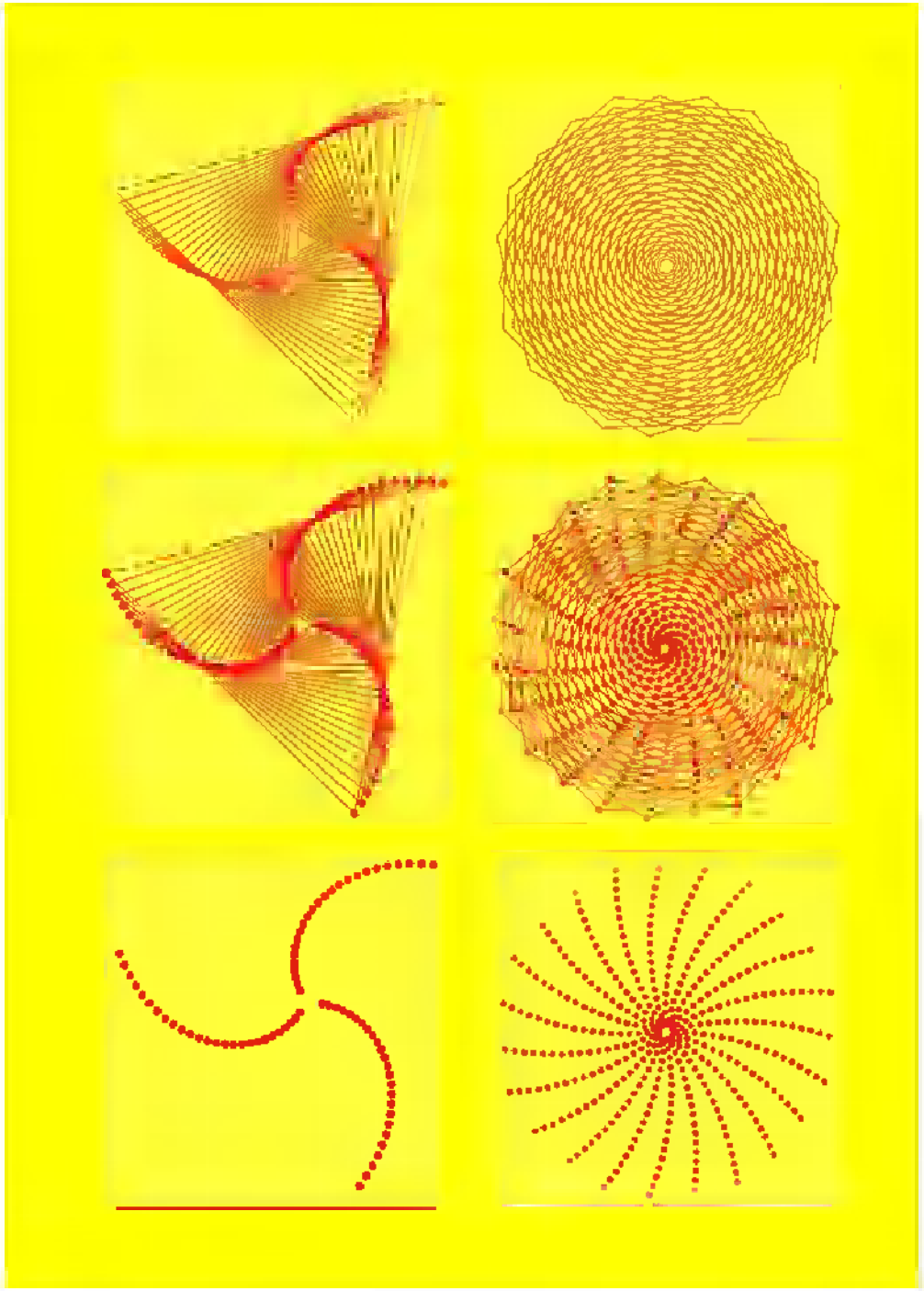


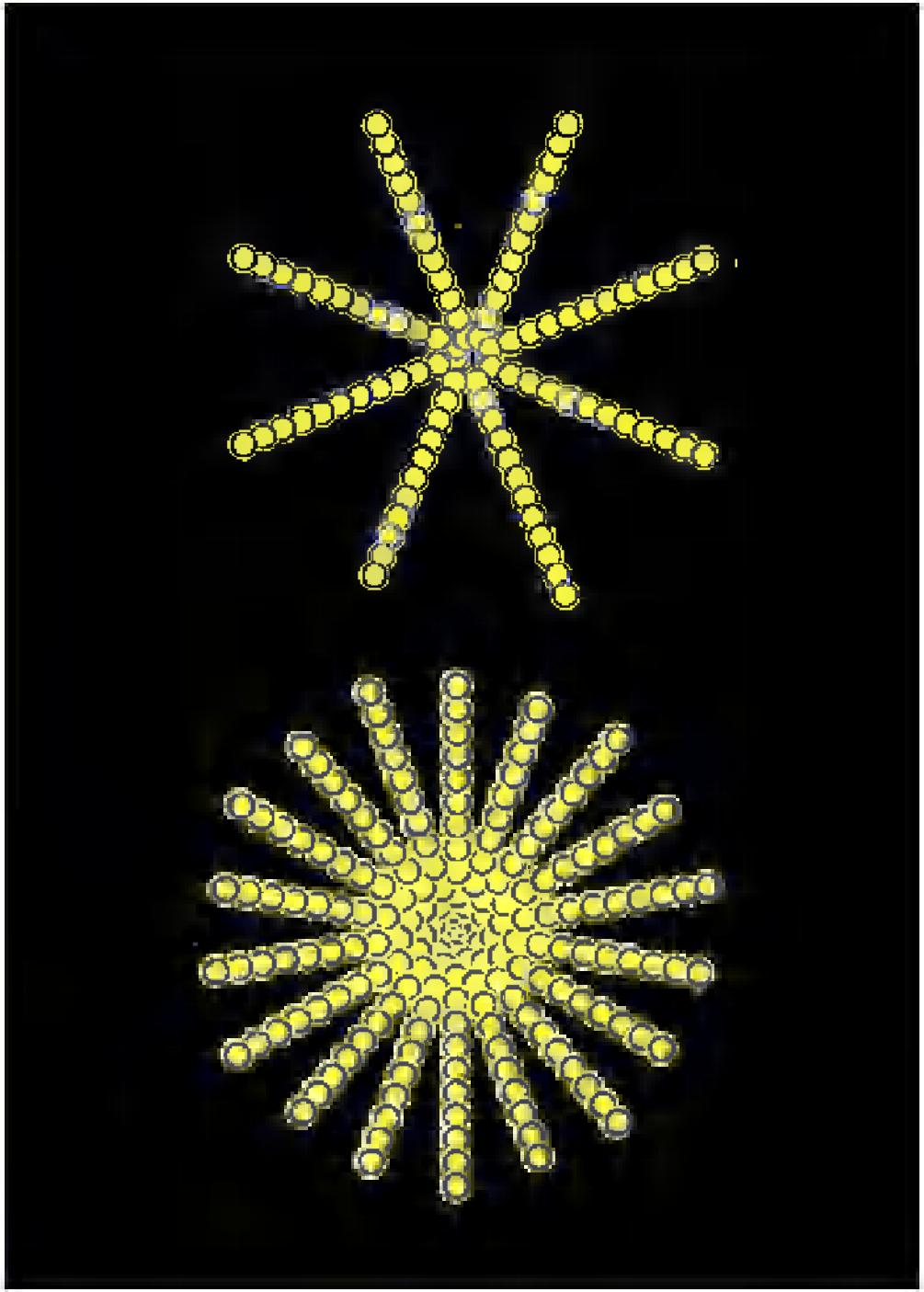
Punktspiralen

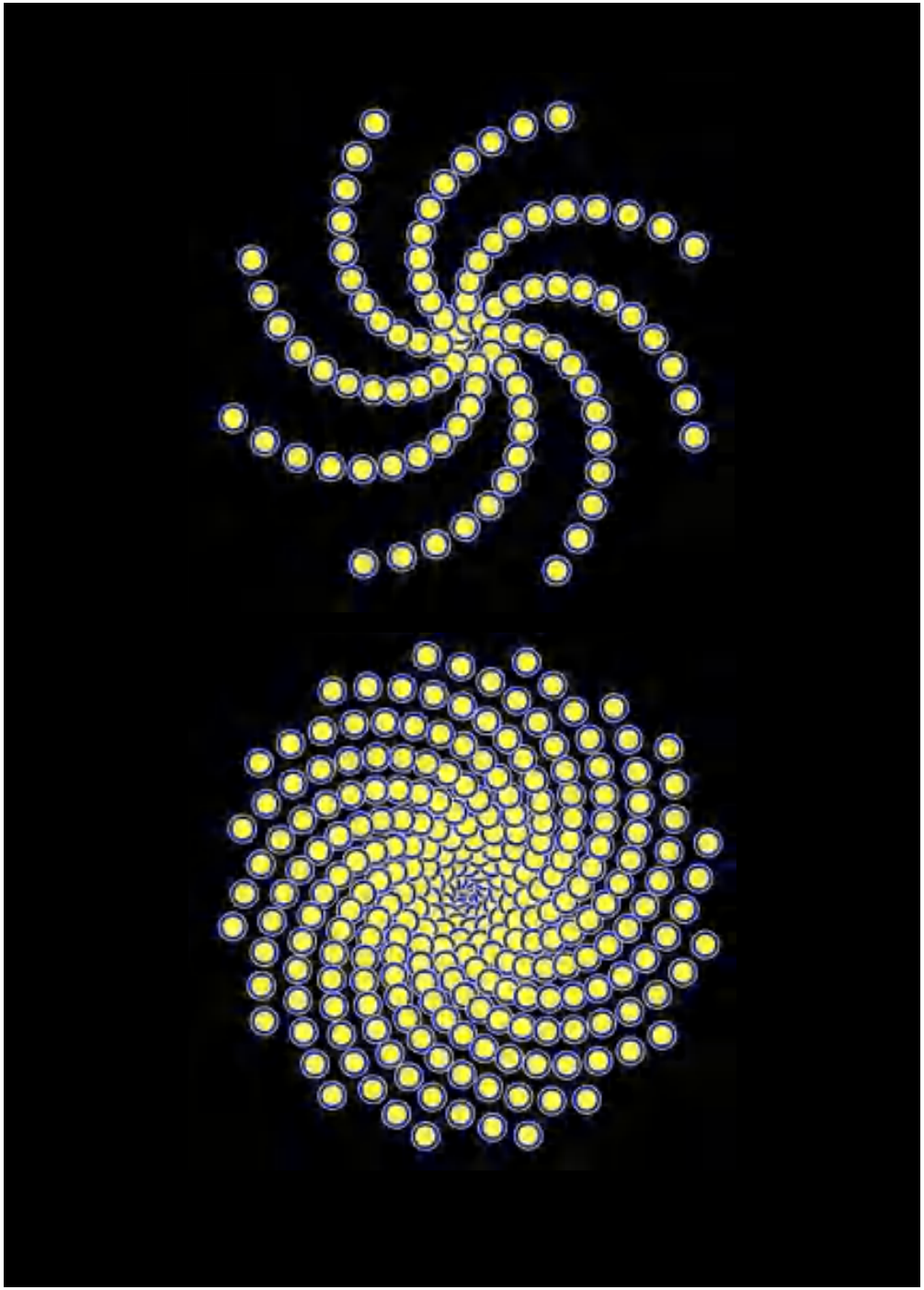
Die Grundstruktur von Spiralen wird oft wesentlich deutlicher sichtbar, wenn nicht (nur) die Strecken zwischen den Spiralpunkten gezeichnet werden, sondern deren Anfangs- und Endpunkte.

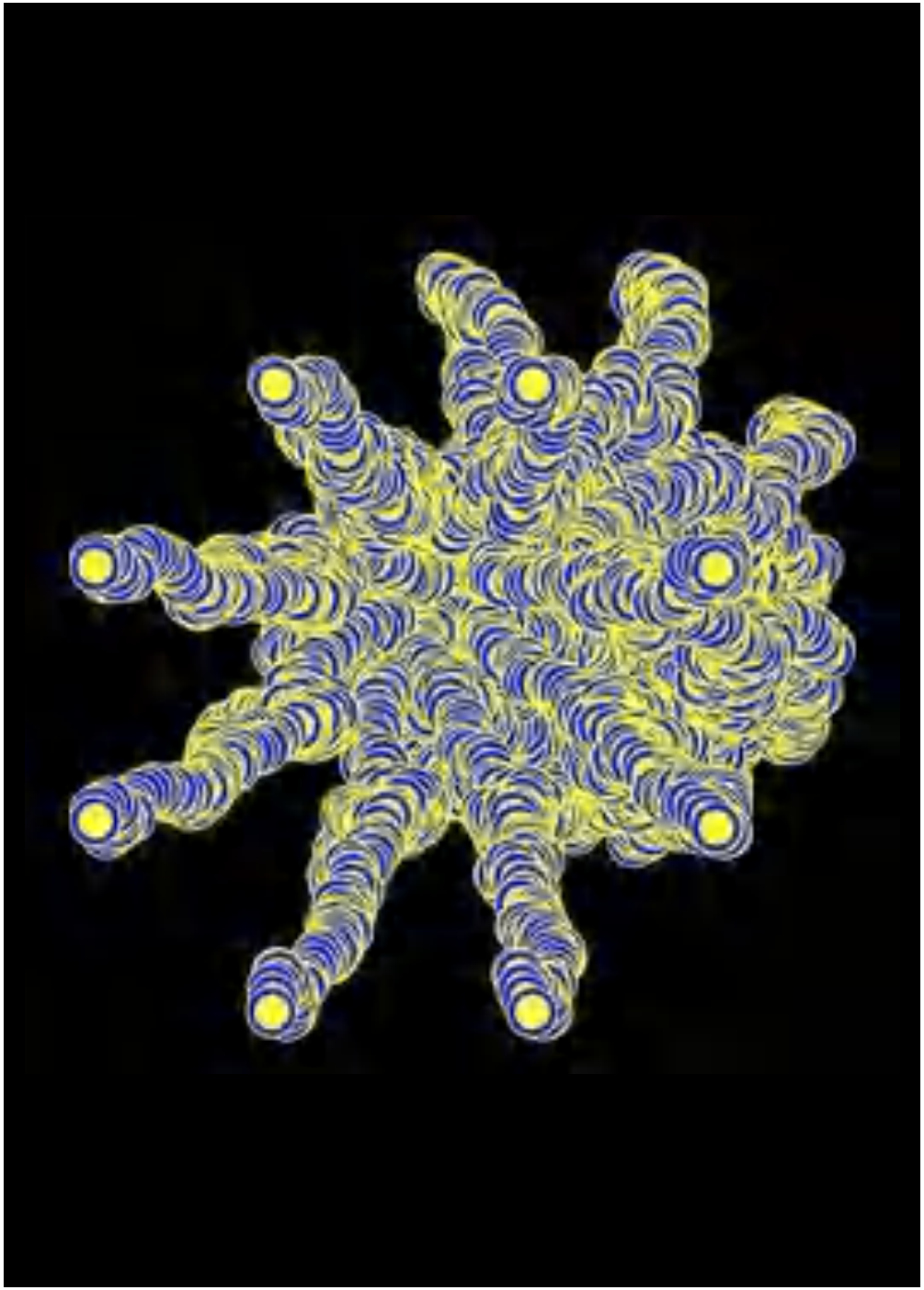
Punkte bestimmter Stiftdicke werden gezeichnet, indem der Schildkröte beim Befehl **gehe Schritte** keine Schrittzahl vorgegeben wird und sie dadurch gewissermaßen „auf der Stelle tritt“. Ein gelber Punkt der Dicke 60 ensteht dann so:

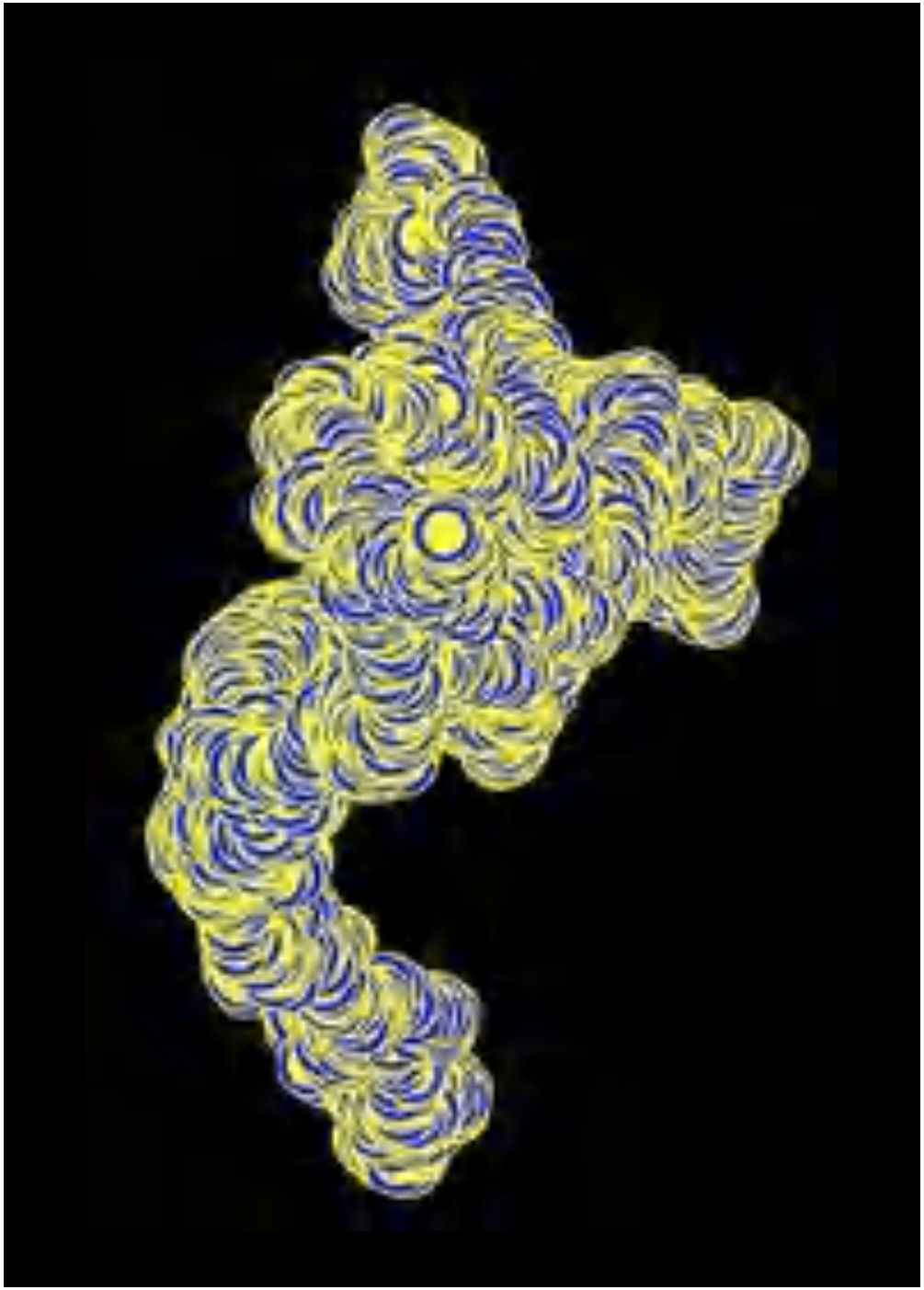
Kleine Veränderungen an den Kenndaten der Spiralen, etwa der **länge**, des **winkel** oder der Änderung **delta**, können starke Änderungen der Punktstrukturen bewirken. Wird dies mit Zufallsänderungen kombiniert, entstehen noch einmal völlig veränderte Strukturen.











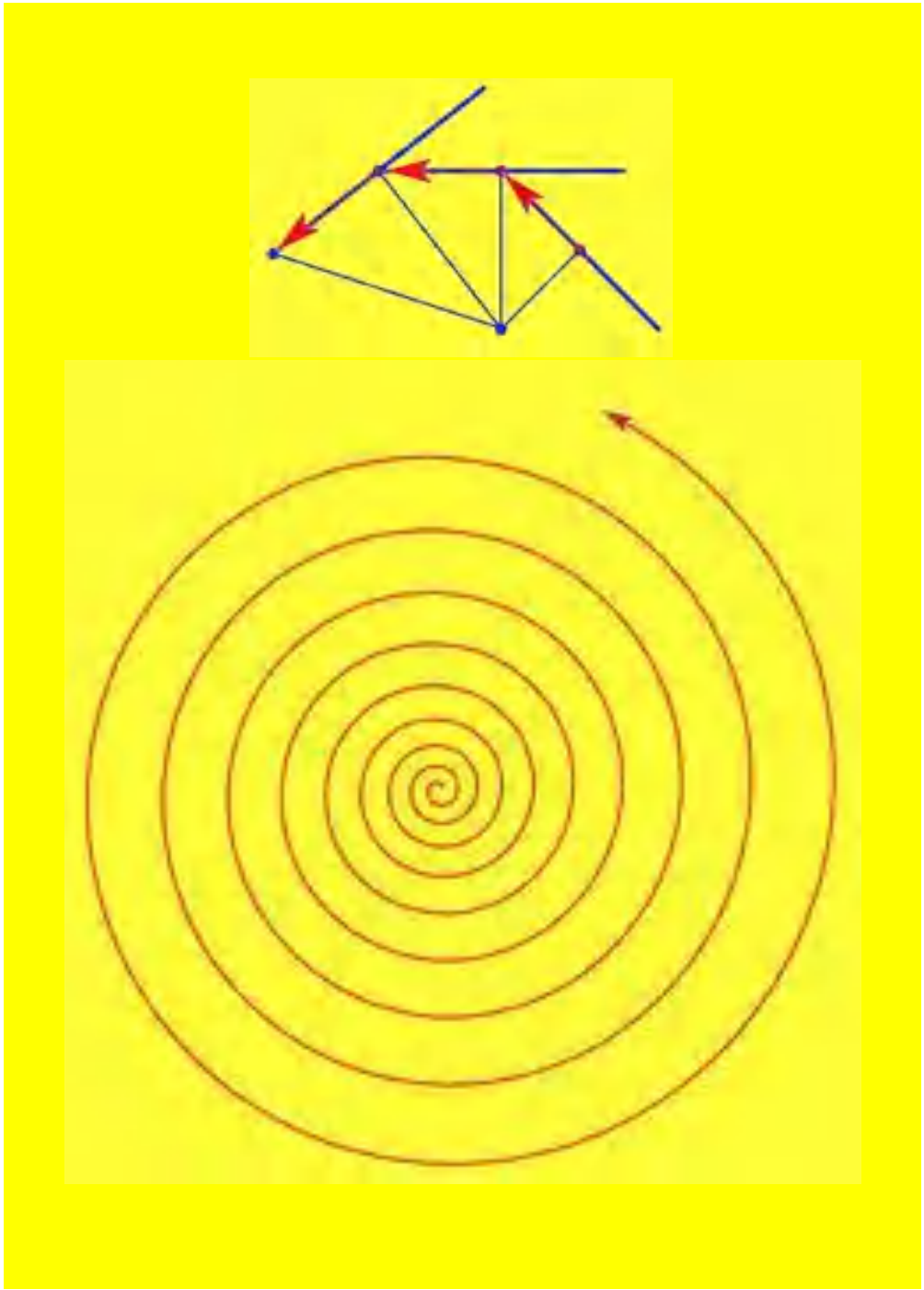
Peilgeometrie

Die Schildkröte besitzt auch **Messfühler** (Sensoren). Damit können unterschiedliche Werte abgefragt und weiterverarbeitet werden:

- **Entfernung zu Mitte/Mauszeiger/Objekt** liefert die Entfernung (in Bildpunkten) von der aktuellen Position der Schildkröte bis zur Bildmitte, bis zum Mauszeiger oder bis zu einem anderen anzugebenden Objekt.
- **Richtung zu Mitte/Mauszeiger/Objekt** liefert die Richtung (in Grad) von der aktuellen Position der Schildkröte zur Bildmitte, zum Mauszeiger oder zu einem anderen anzugebenden Objekt.
- Genauso können weitere Eigenschaften erkannt und abgefragt werden. So liefert etwa **Farbton, Sättigung ... bei** die Farbwerte unterhalb des Mauszeigers oder einem anderen anzugebenden Objekt.

Das kann genutzt werden, um Spiralen auf eine ganz andere Art zu erzeugen; es wird dabei von **Peilgeometrie** gesprochen.

Ausgangsposition ist ein Punkt auf einem Kreis um einen gegebenen Mittelpunkt (**blau** im Bild rechts oben). Für diesen Punkt kann die **Kreistangente** gezeichnet werden (senkrecht zur Abstandslinie zum Mittelpunkt). Auf dieser Tangente wird die Schildkröte um **länge** bewegt. An ihrer neuen Position wird wieder die Tangente gezeichnet und darauf die Schildkröte um **länge + delta** bewegt.



Rekursive Bäume

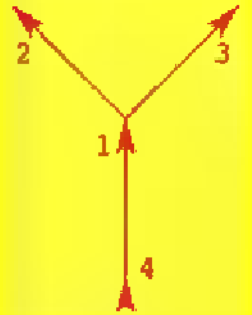
Das **Zeichnen von Bäumen** ist ein beliebtes Beispiel zur Anwendung von **Rekursion** in der Schildkötengrafik. Ein Grund dafür ist, dass beim Zeichnen eines Baums immer wiederkehrende Abläufe zu finden sind:

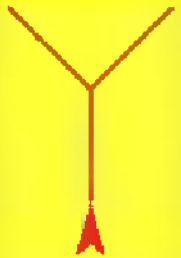
1. Zuerst wird der Stamm gezeichnet.
2. An einer Verzeigungsstelle wird ein linker Ast gezeichnet.
3. An derselben Verzeigungsstelle wird ein rechter Ast gezeichnet.
4. Wichtig: Am Ende kehrt die Schildkröte zum Ausgangspunkt zurück.

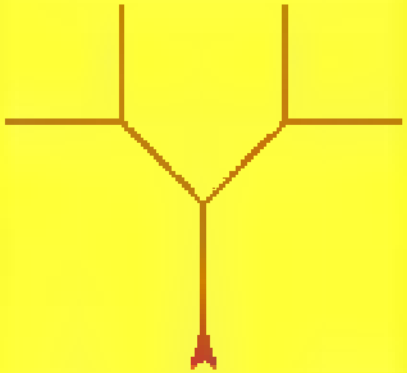
Werden die Äste jeweils als Ausgangspunkt weiterer Verästelungen verwendet, entsteht eine komplexe Verzweigungsstruktur, also der Baum mit Astwerk.

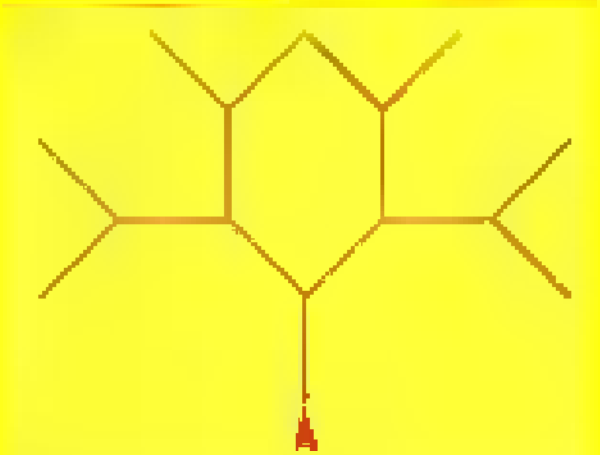
In der rekursiven Fassung wird deshalb **gehe länge Schritte** beim Zeichnen der Äste ersetzt durch den rekursiven Aufruf **baum_rekursiv länge tiefe**. Im Bild rechts sind die Ergebnisse für eine Tiefe von 1 bis 4 dargestellt. Wenn **tiefe = 0** den Wert **wahr** liefert, wird das Zeichnen abgebrochen.

Auch die **länge** und der Winkel, mit dem die Ästa abzweigen, können abhängig von der **tiefe** verändert werden. Kombiniert mit Strichdicke und Farbe ähneln die Ergebnisse dann bereits sehr realen Bäumen bestimmter Arten. Andere Werte ergeben abstrakte Figuren mit flächenfüllenden Mustern.

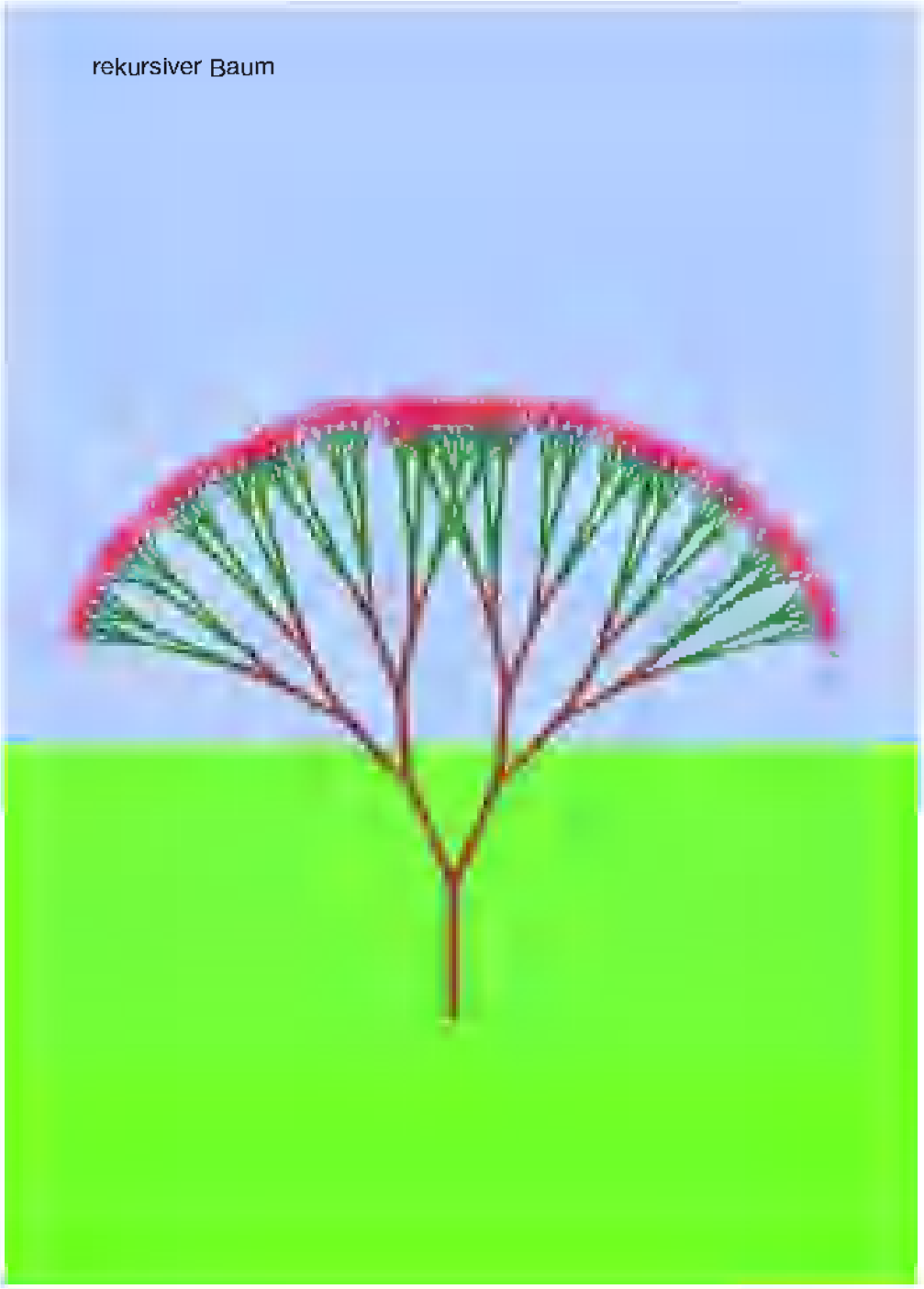




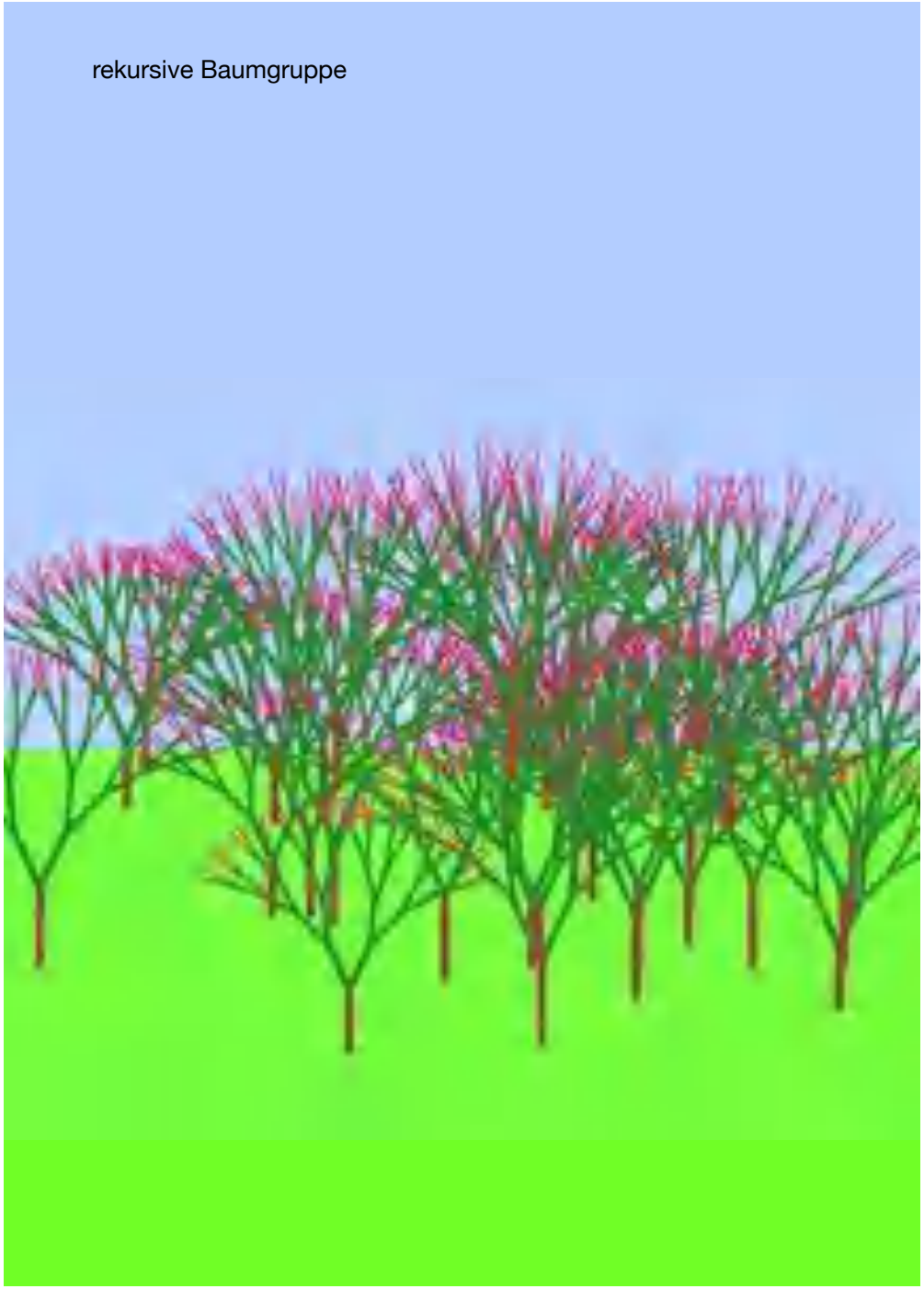




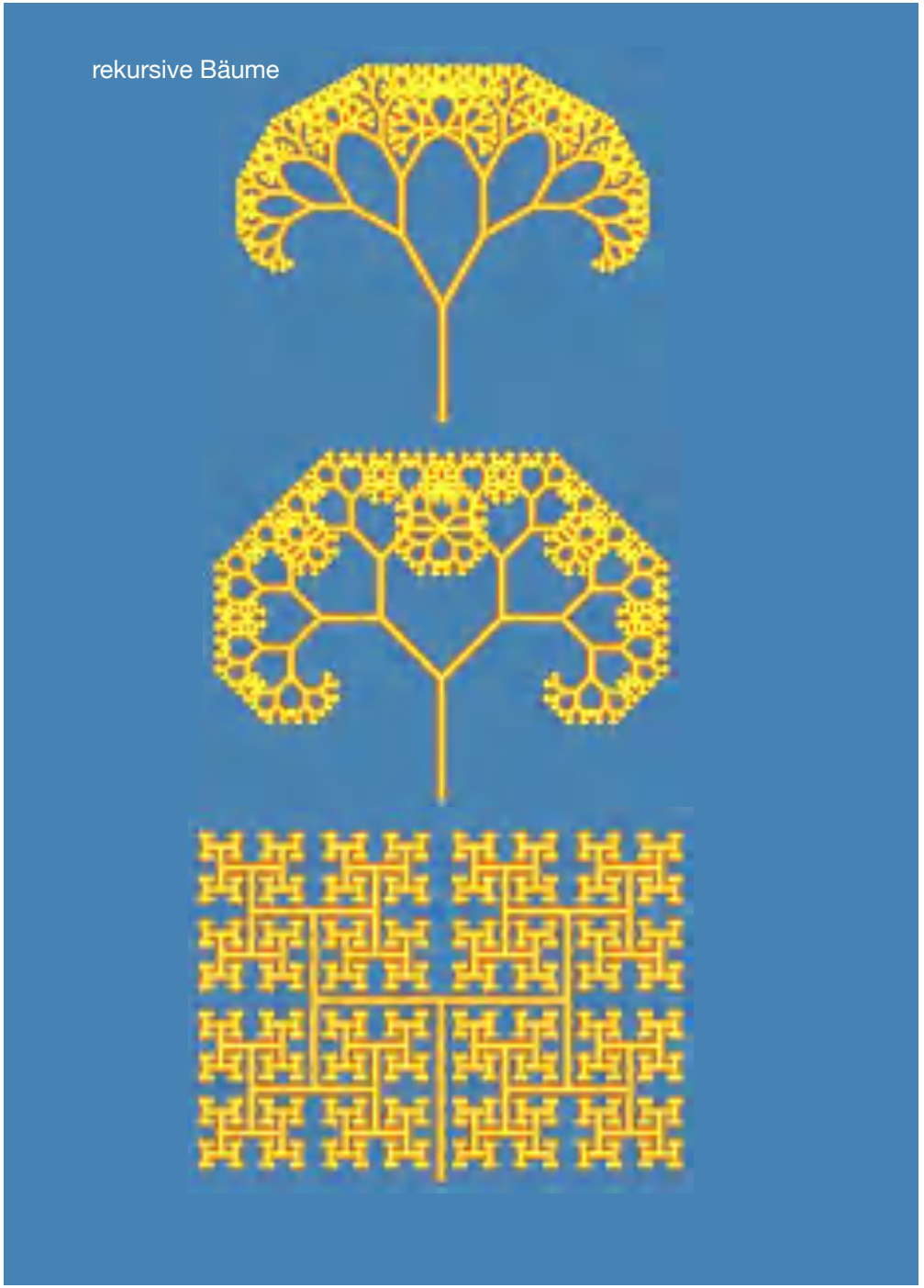
rekursiver Baum

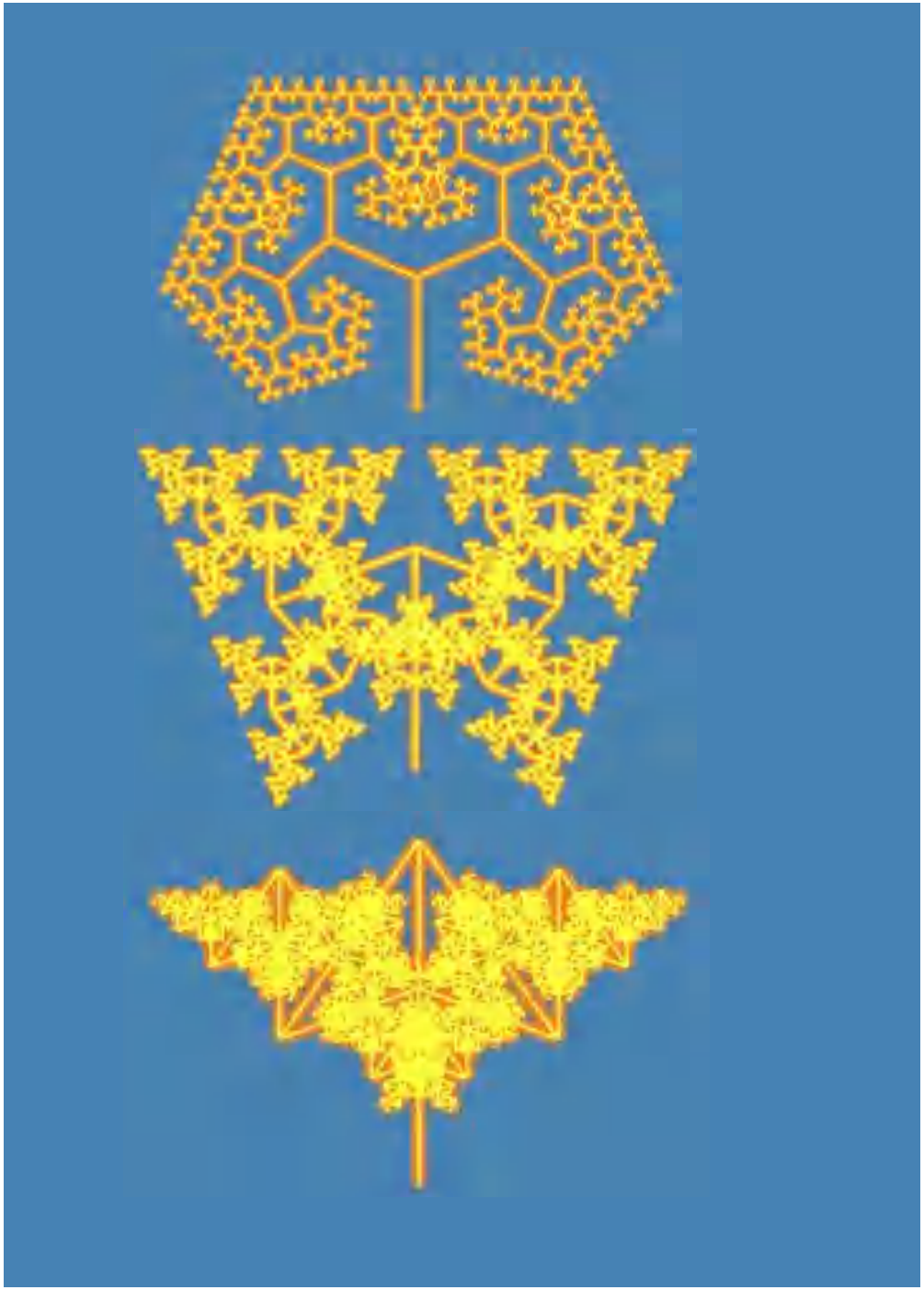


rekursive Baumgruppe



rekursive Bäume



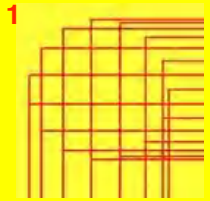


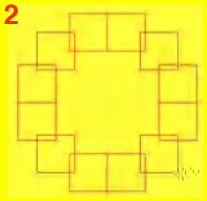
Kombination von Bekanntem

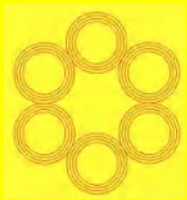
Mit den bisher gezeigten Programmen und Prozeduren konnten Punkte, Linien, Vielecke, Kreise, Kreisbögen, Spiralen und rekursive Bäume gezeichnet werden. Damit steht bereits ein leistungsfähiger **Figurenbaukasten** zur Verfügung. In einem weiteren Schritt können diese Elemente miteinander kombiniert werden[7]. Dauaus ergeben sich wieder neue Strukturen:

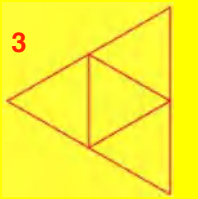
1. Linien auf Kreisen
2. Polygone auf Kreisen
3. Polygonkombinationen
4. Kreiskombinationen
5. rekursive Dreiecke, Quadrate, Polygone
6. rekursive Polygonkombinationen
7. rekursive Kreise

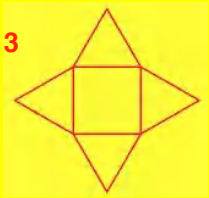
[7] Da also bekannte Elemente verwendet werden, verzichte ich in diesem Abschnitt auf den Abdruck von Code-Schnipseln.

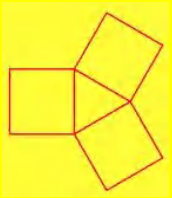


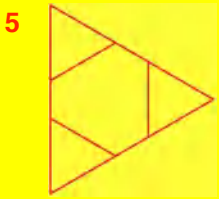


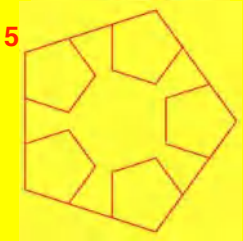


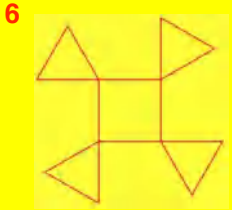


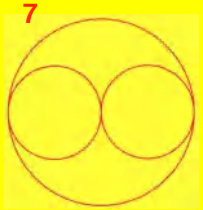


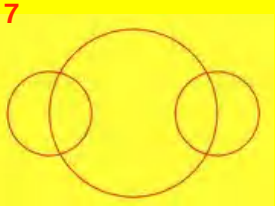




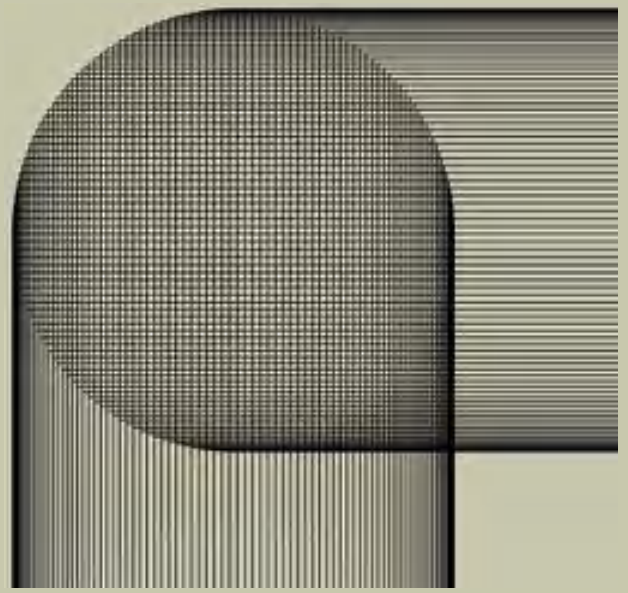


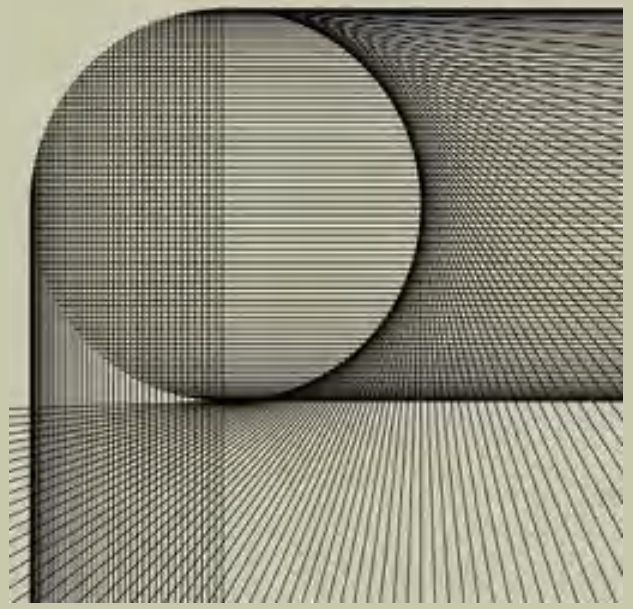


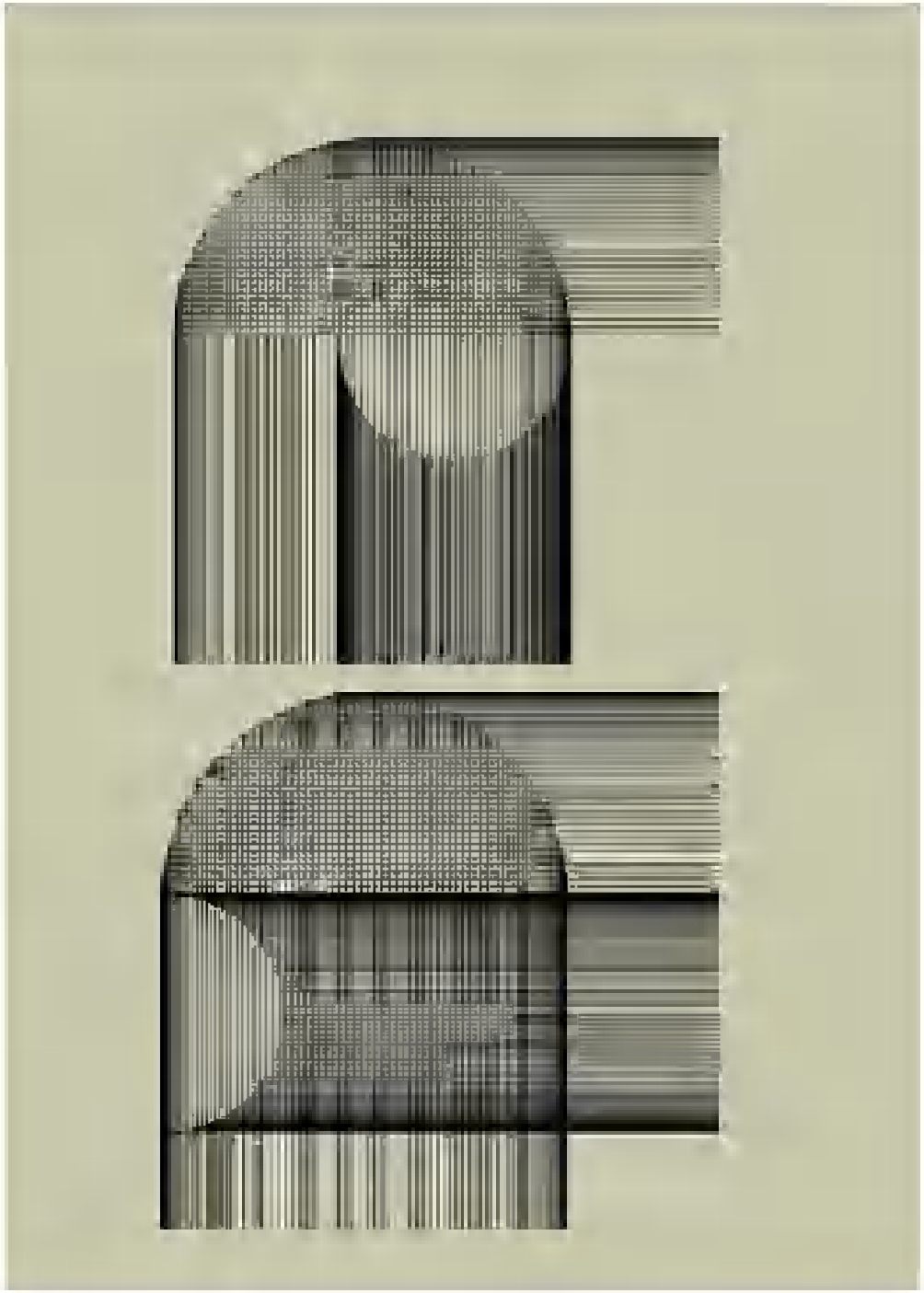


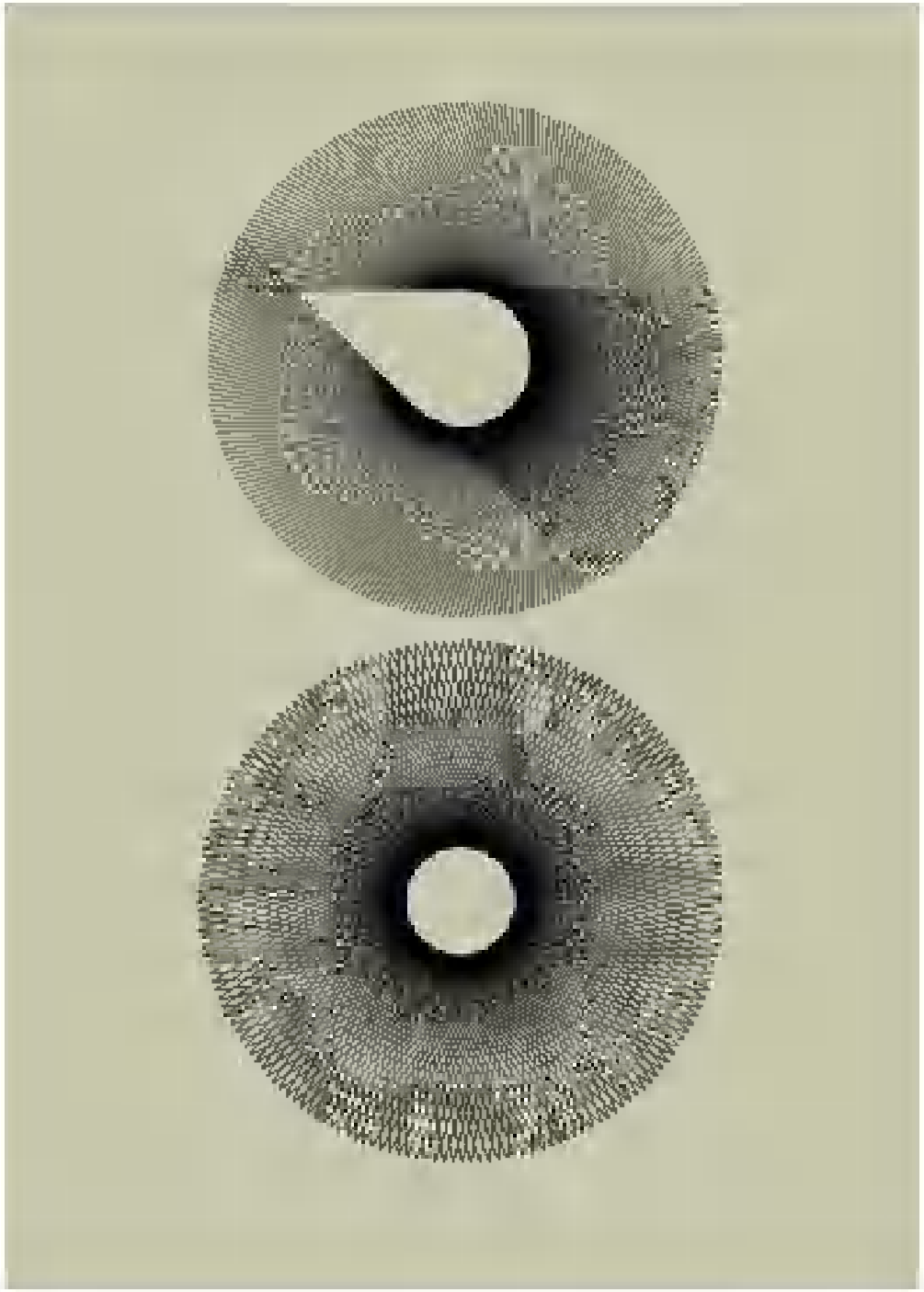


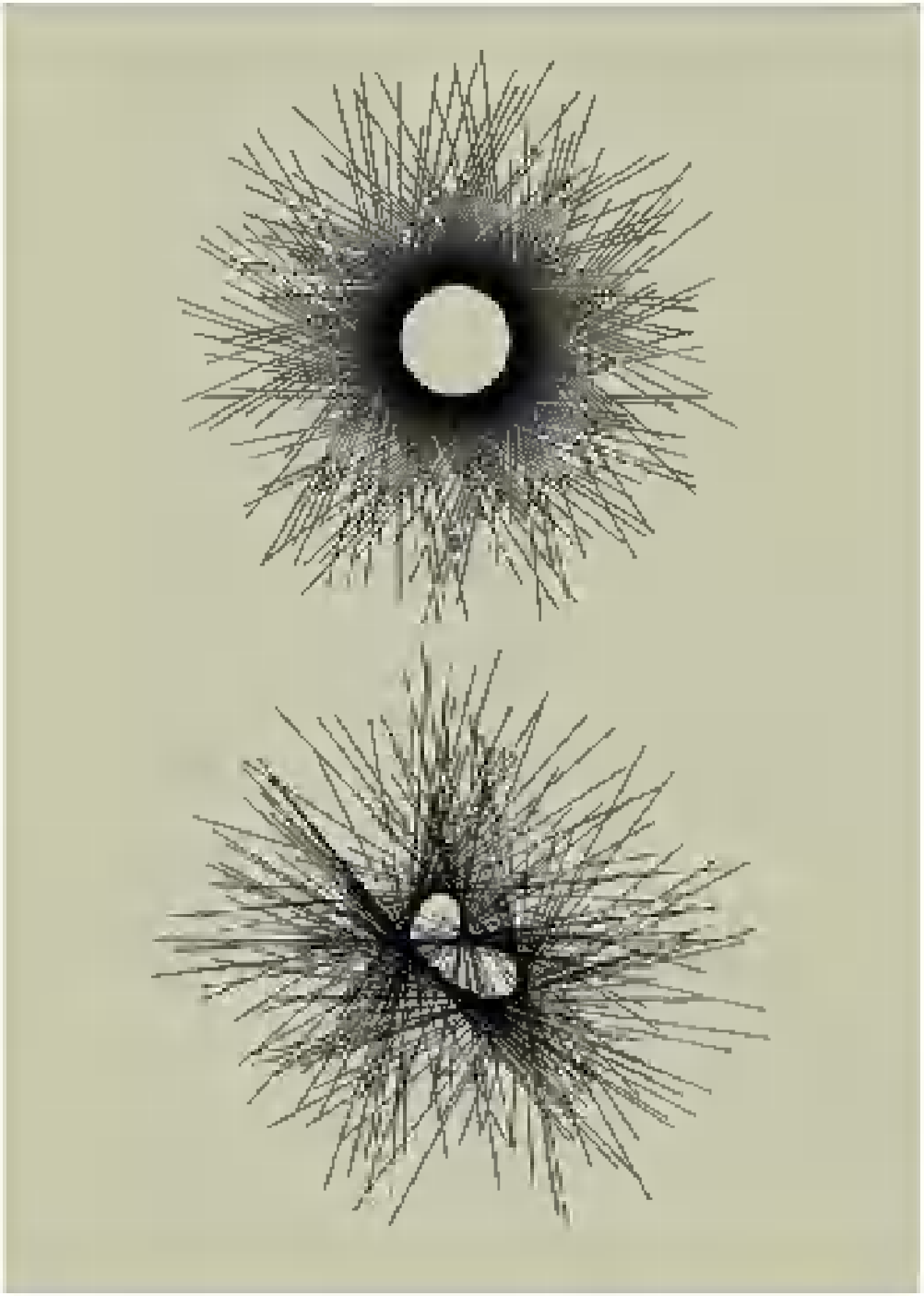
Linien auf Kreis



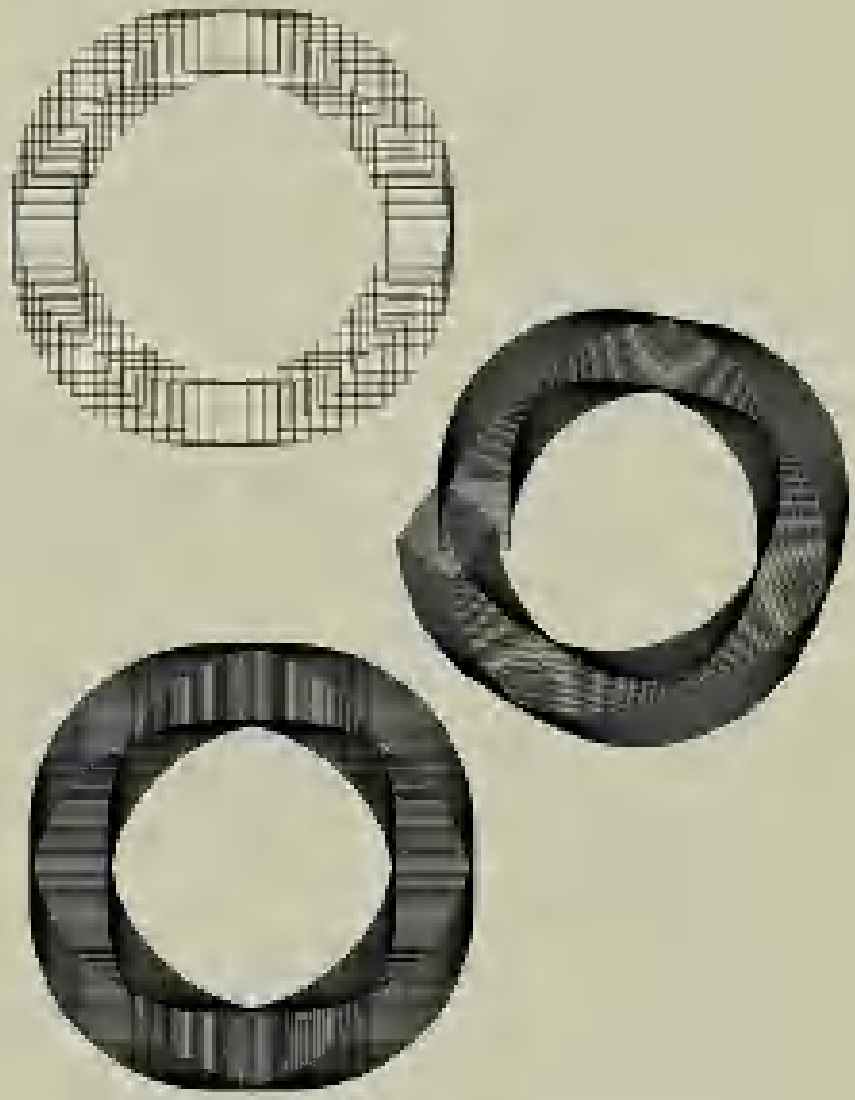


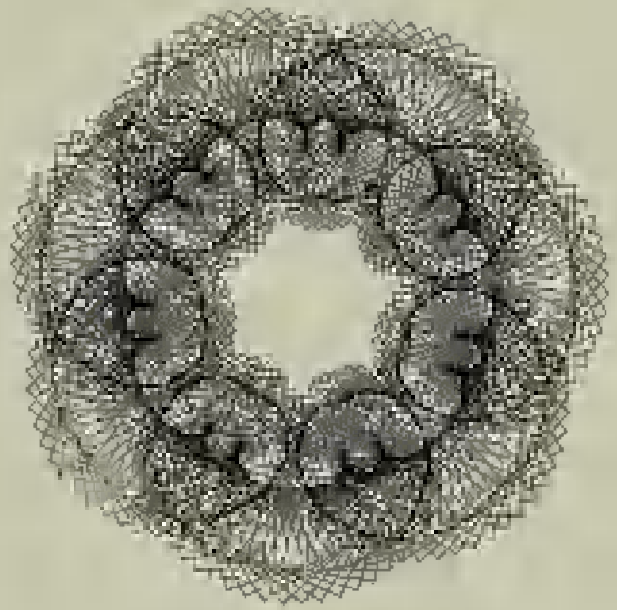




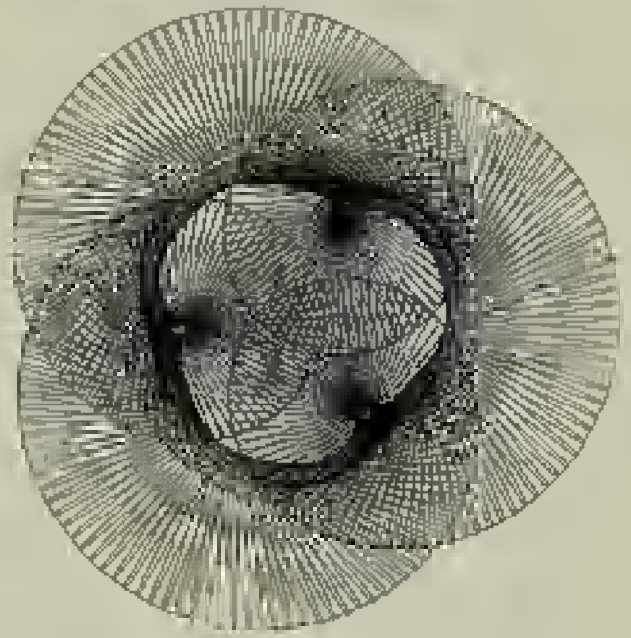


Quadrate auf Kreis

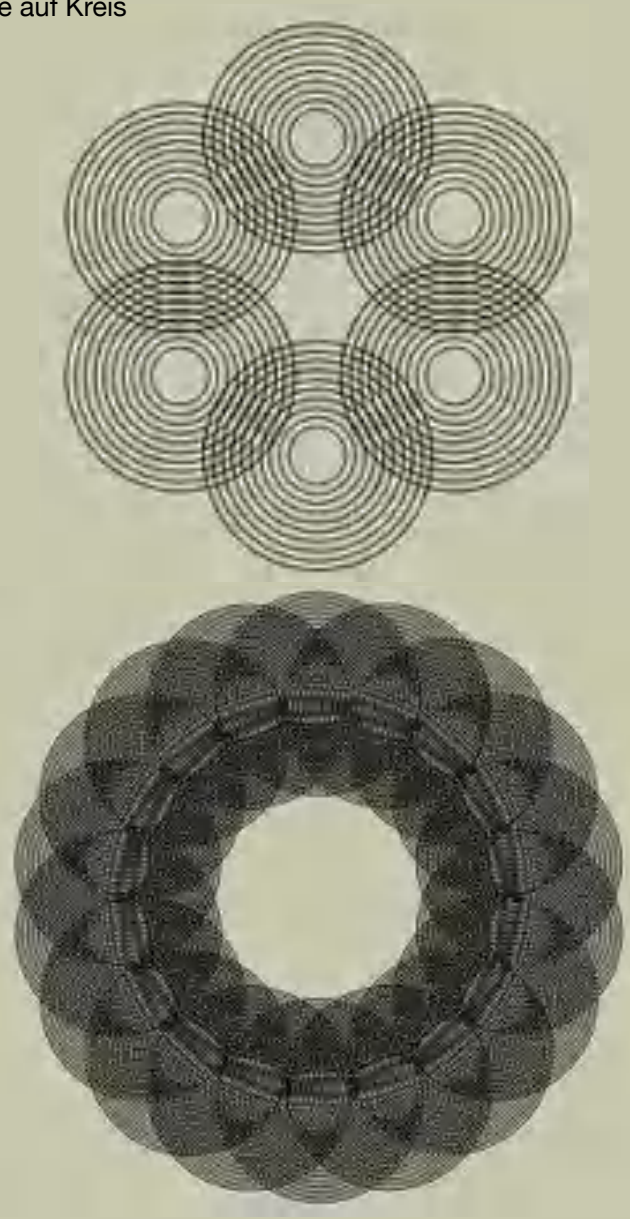




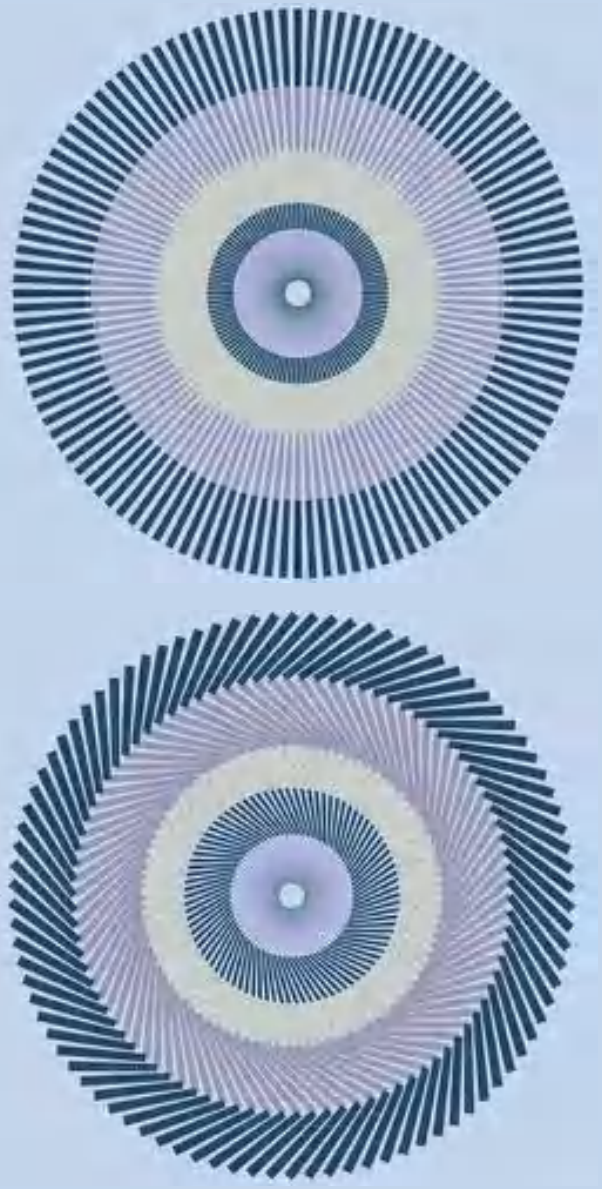
Rechtecke auf Kreis

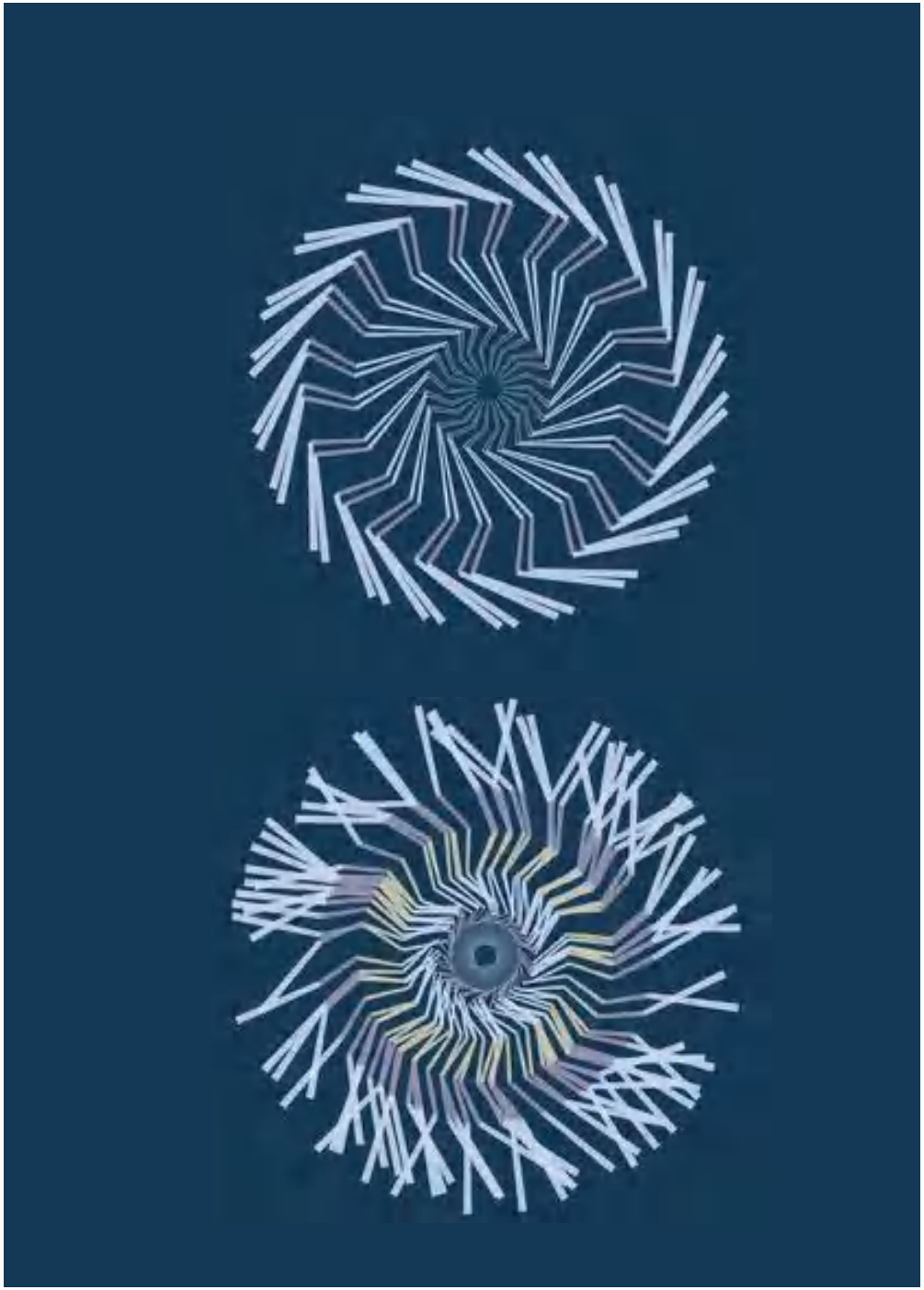


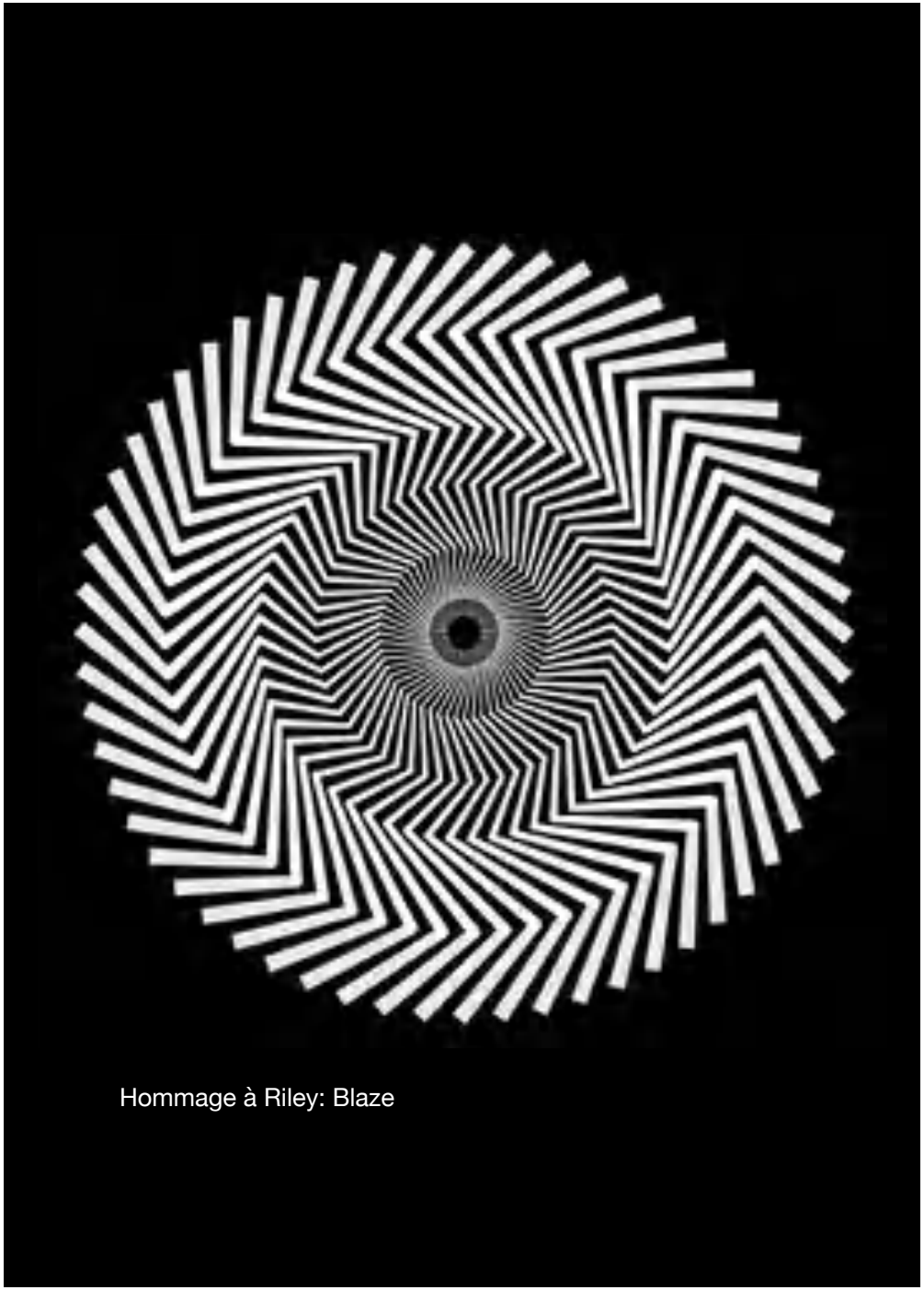
Kreise auf Kreis



Linien auf Kreisen

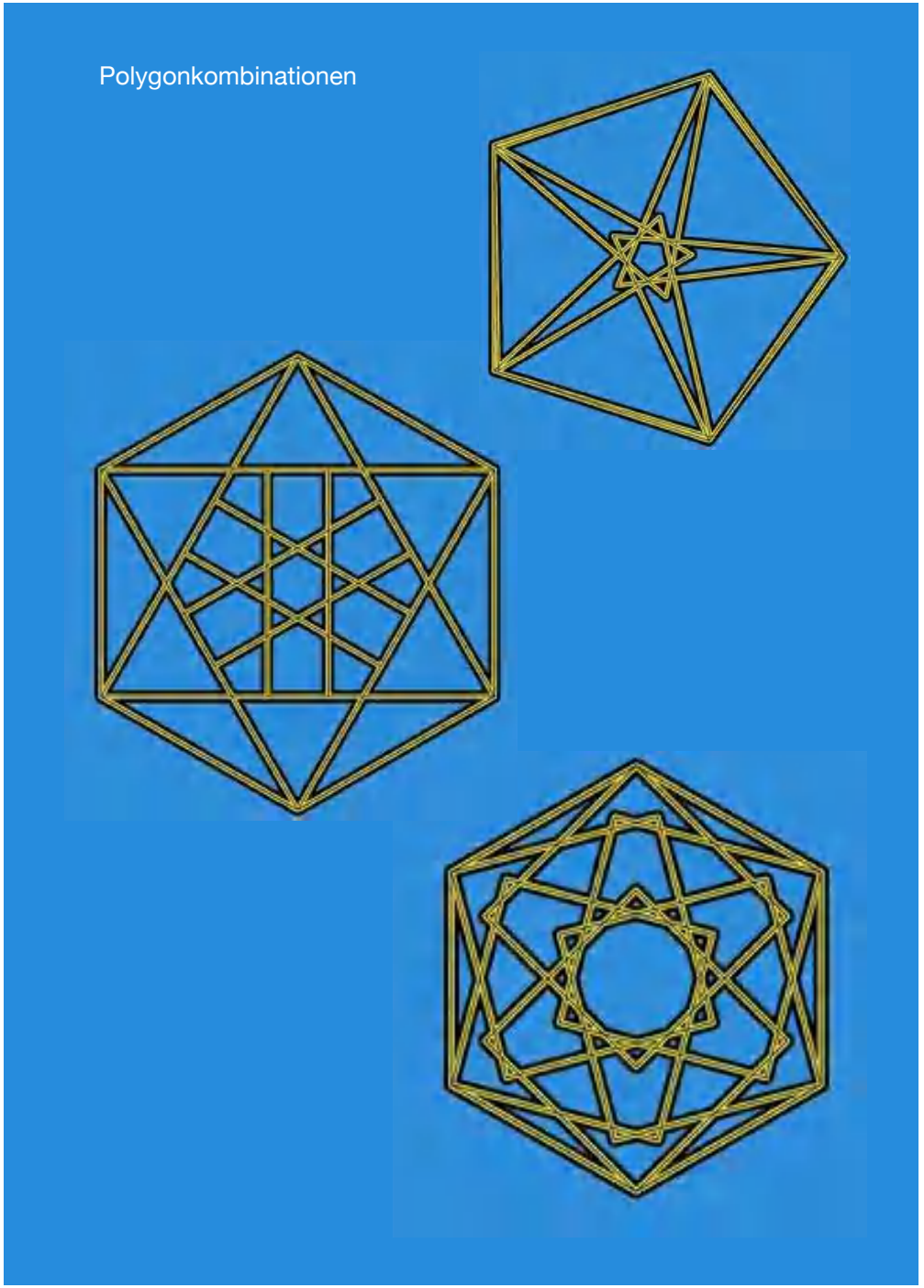


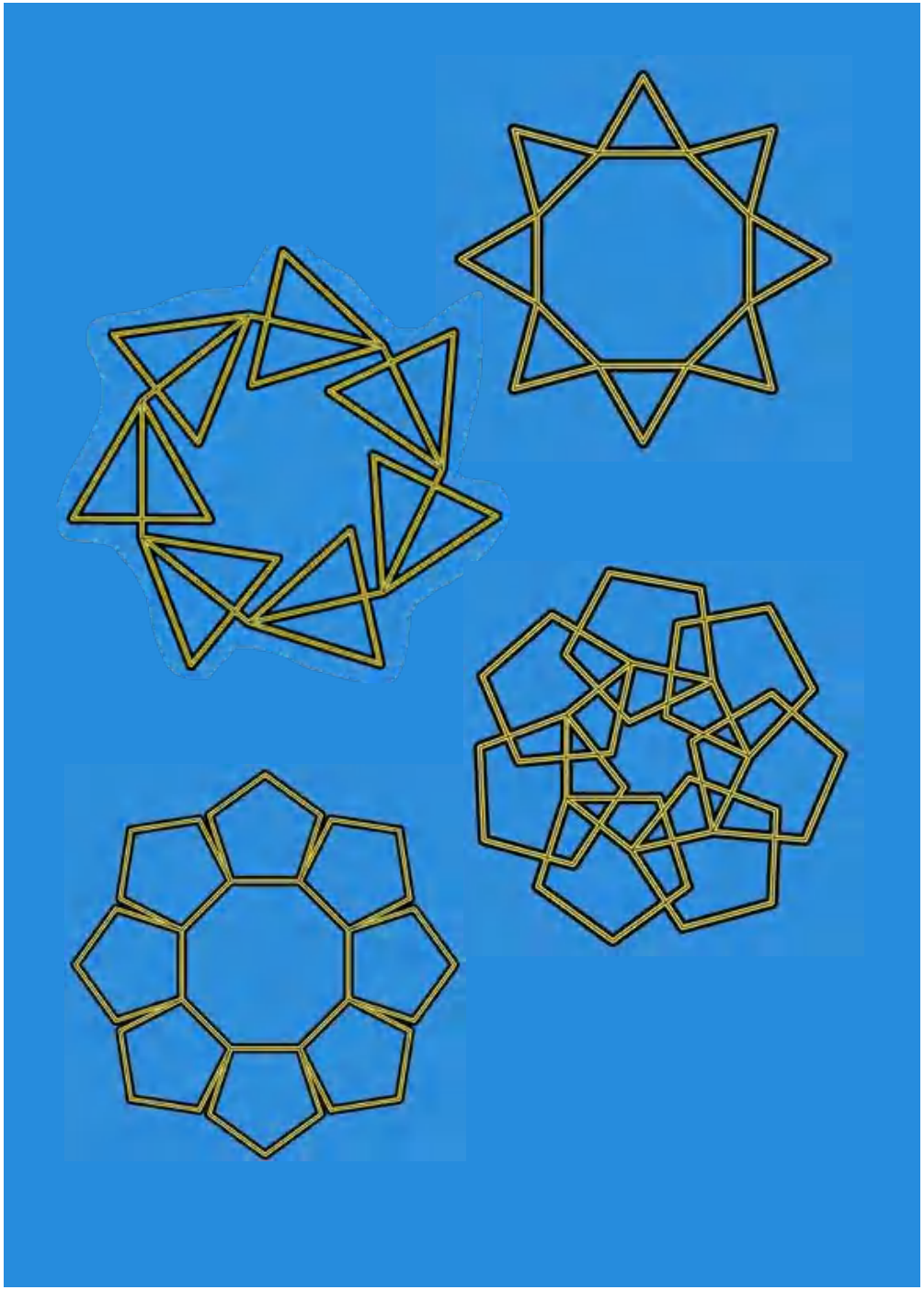




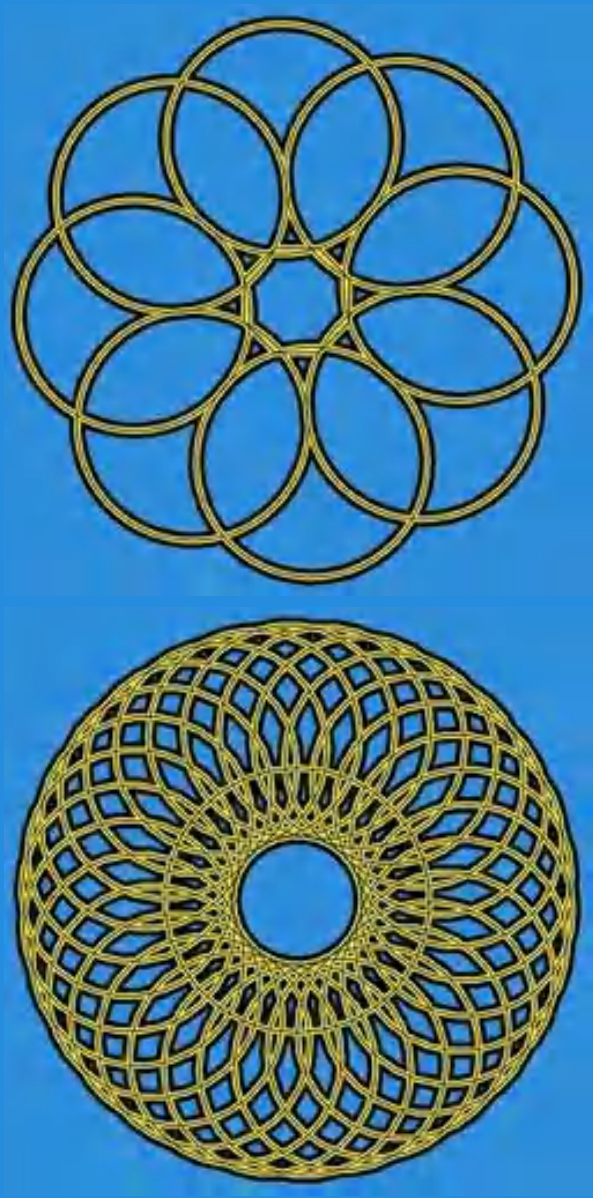
Hommage à Riley: Blaze

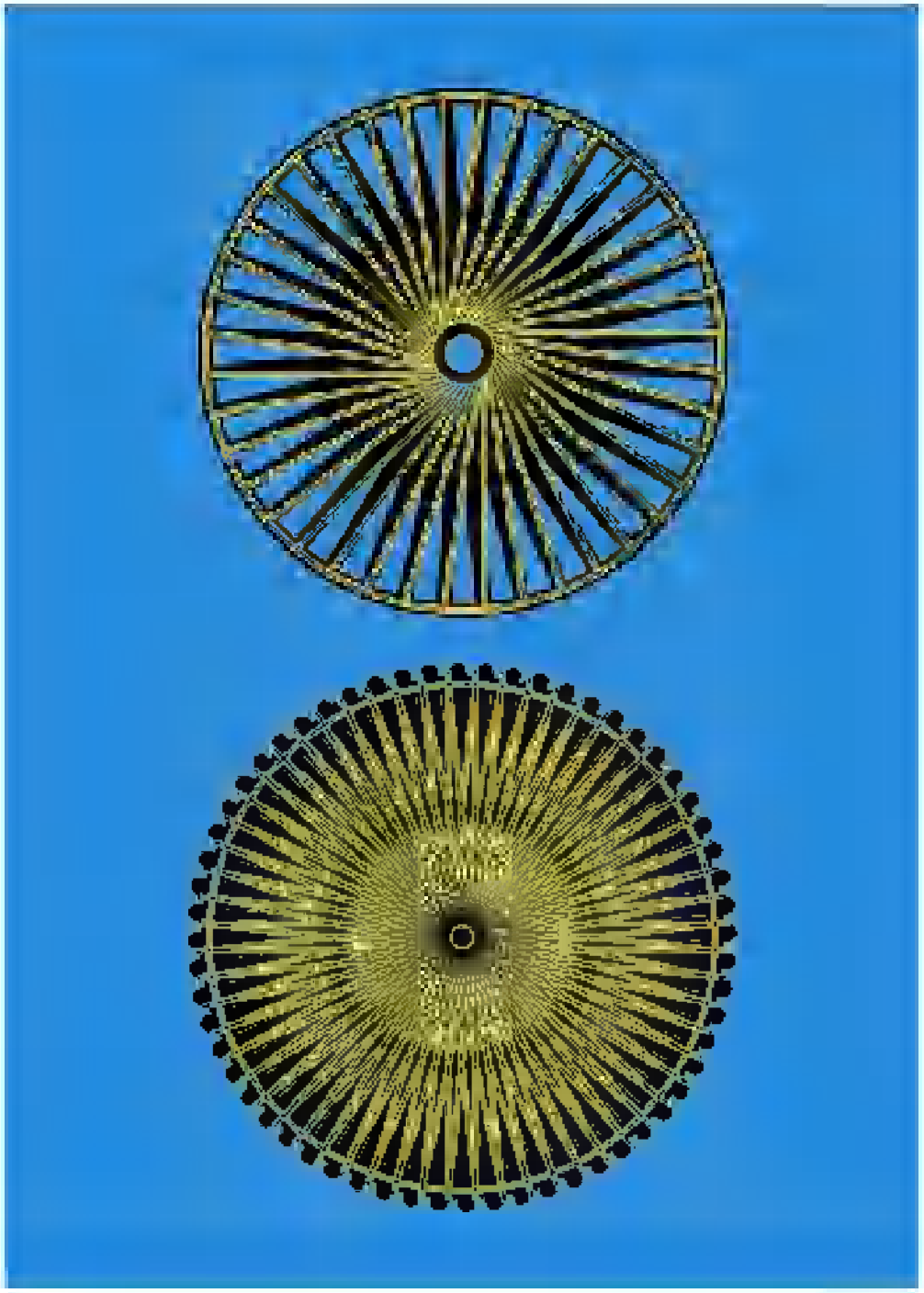
Polygonkombinationen

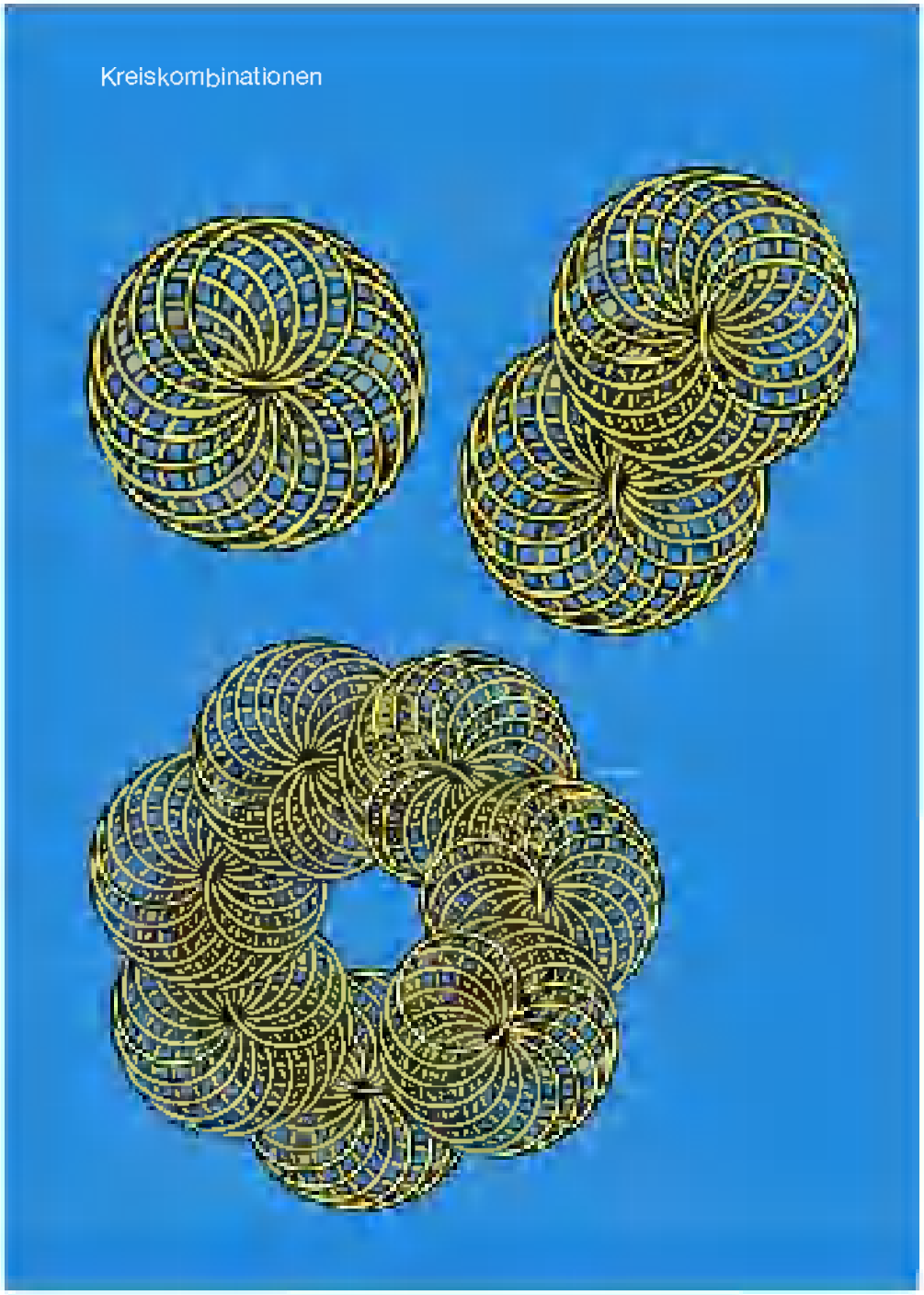


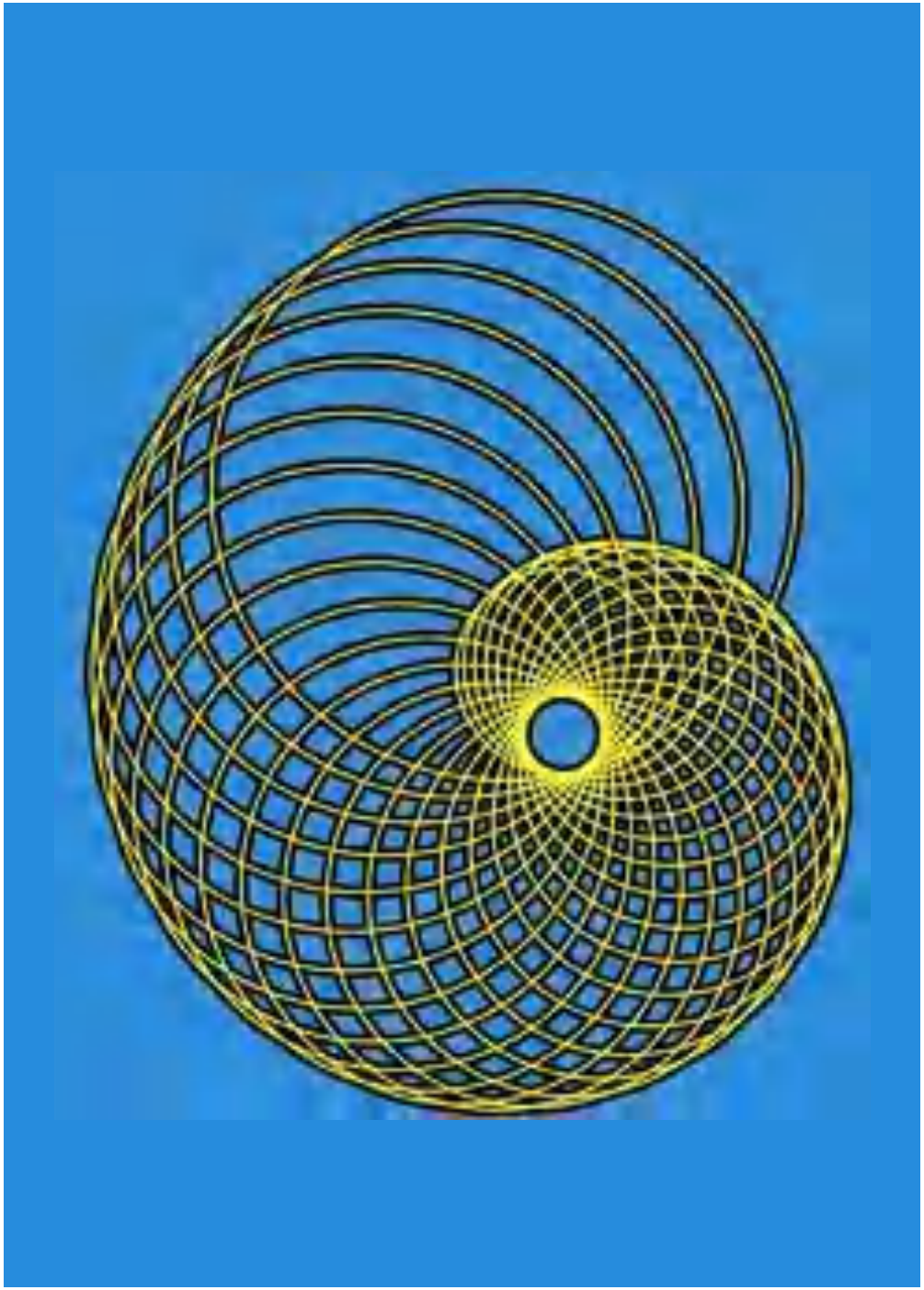


Polygonkombinationen

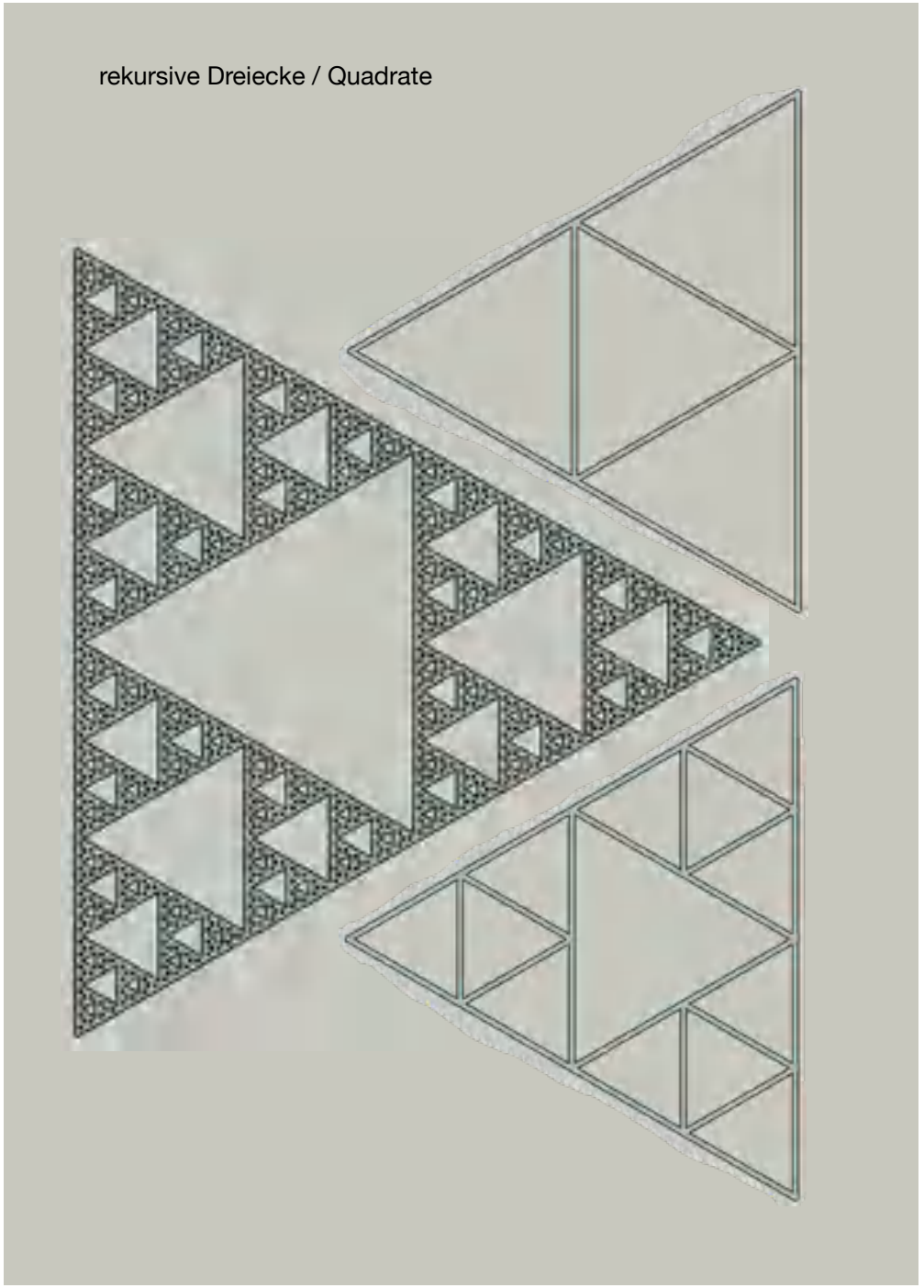


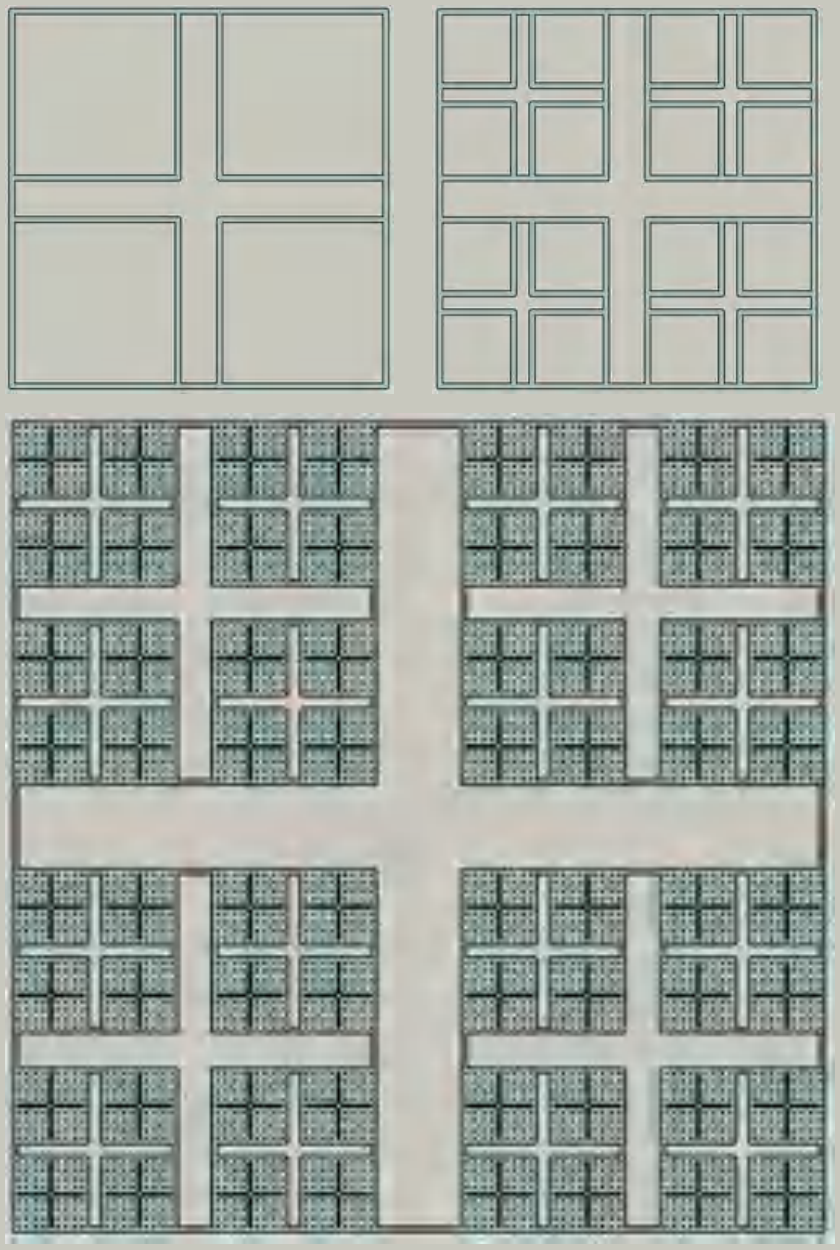



Kreiskombinationen

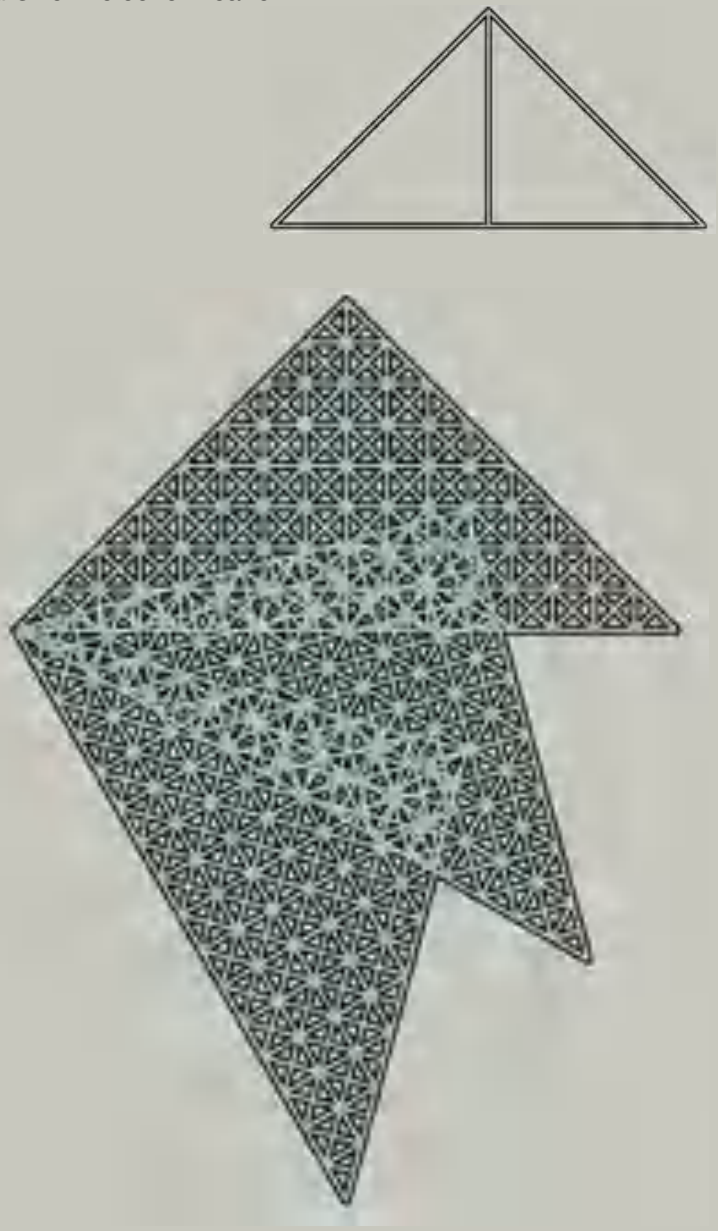


rekursive Dreiecke / Quadrate

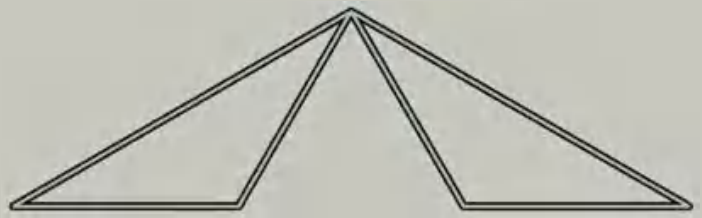


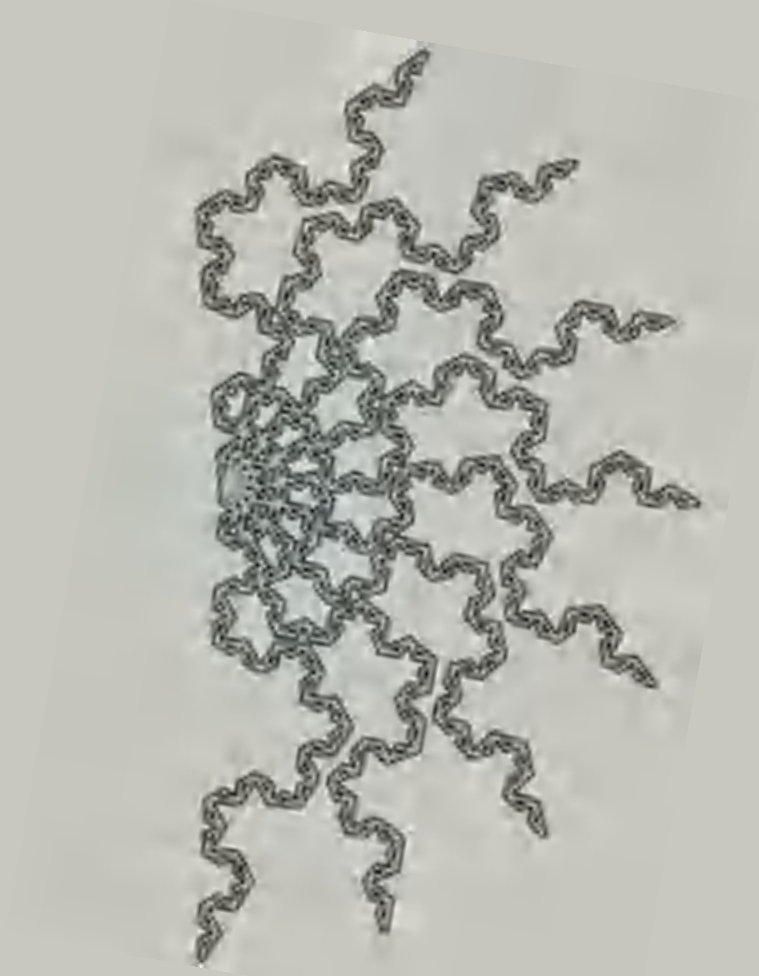


rekursive Dreiecke: Peano

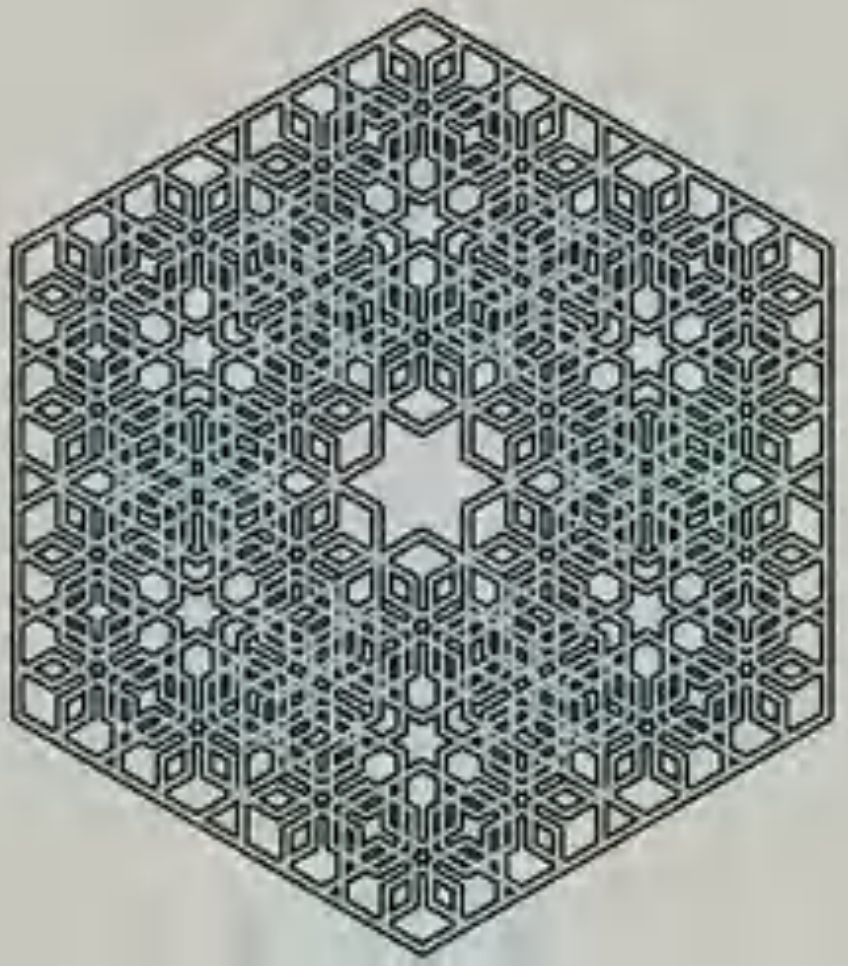


rekursive Dreiecke: Koch

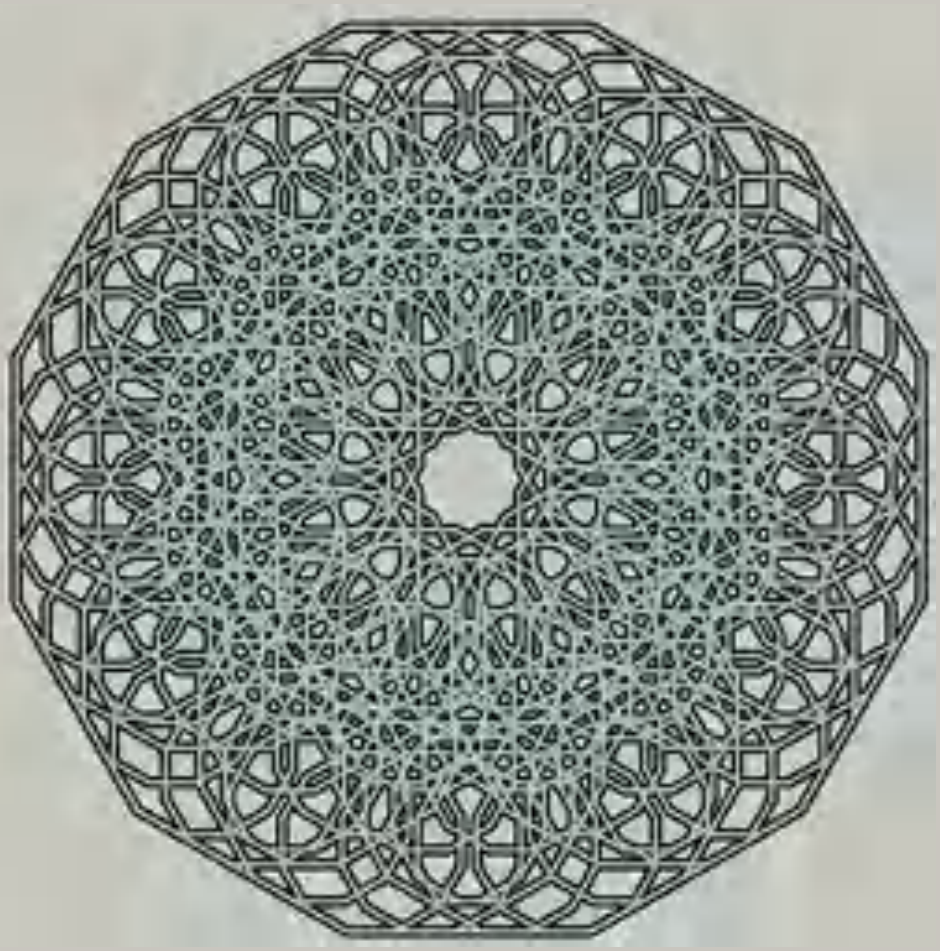




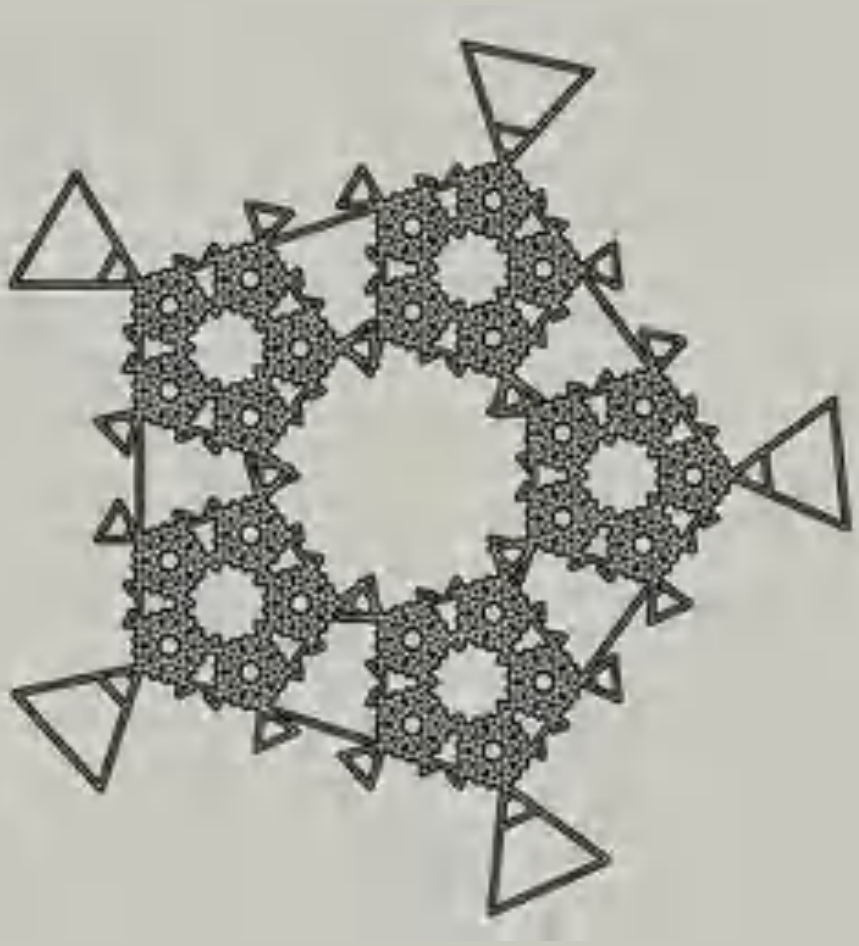
rekursive Sechsecke

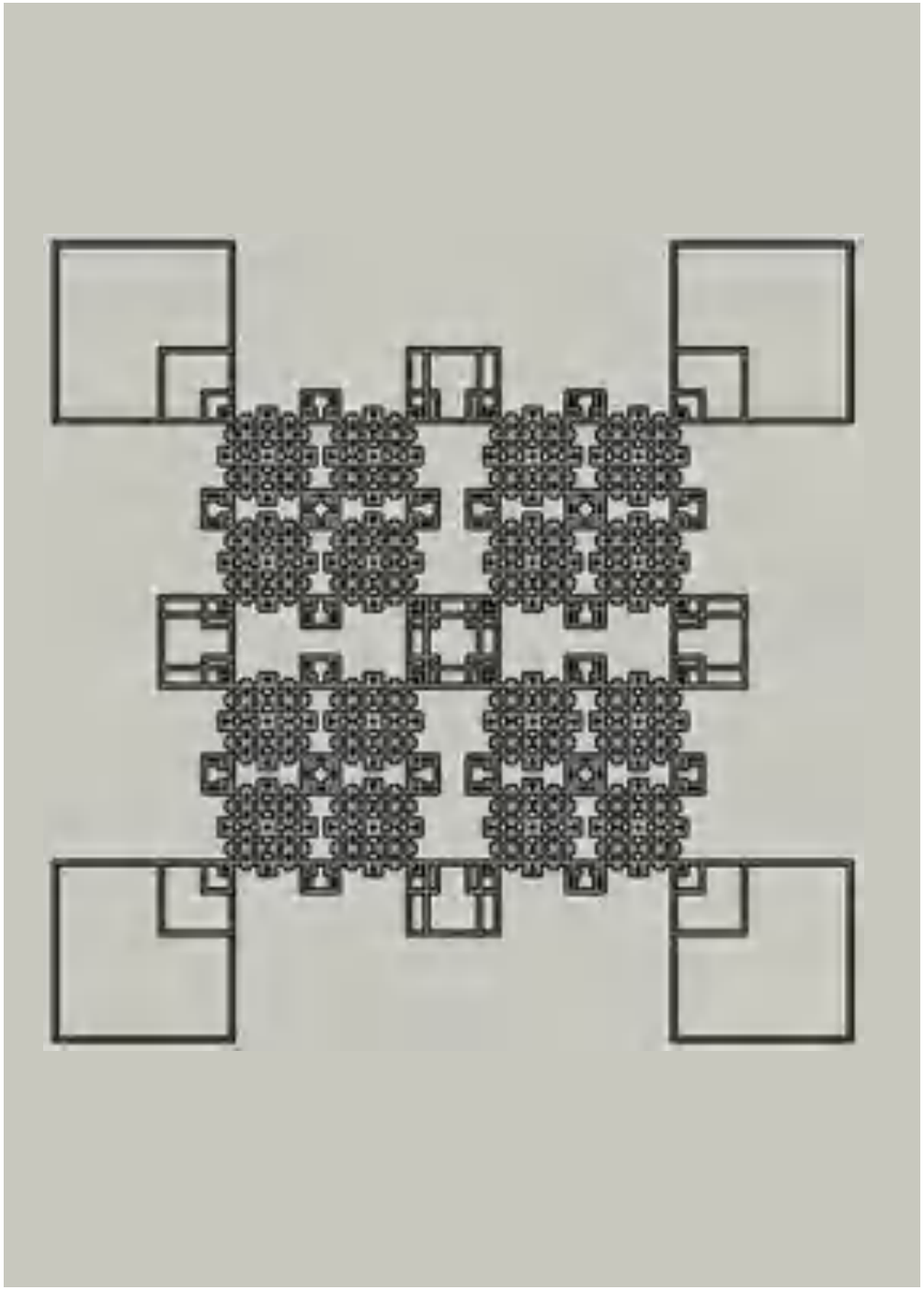


rekursive Zwölfecke

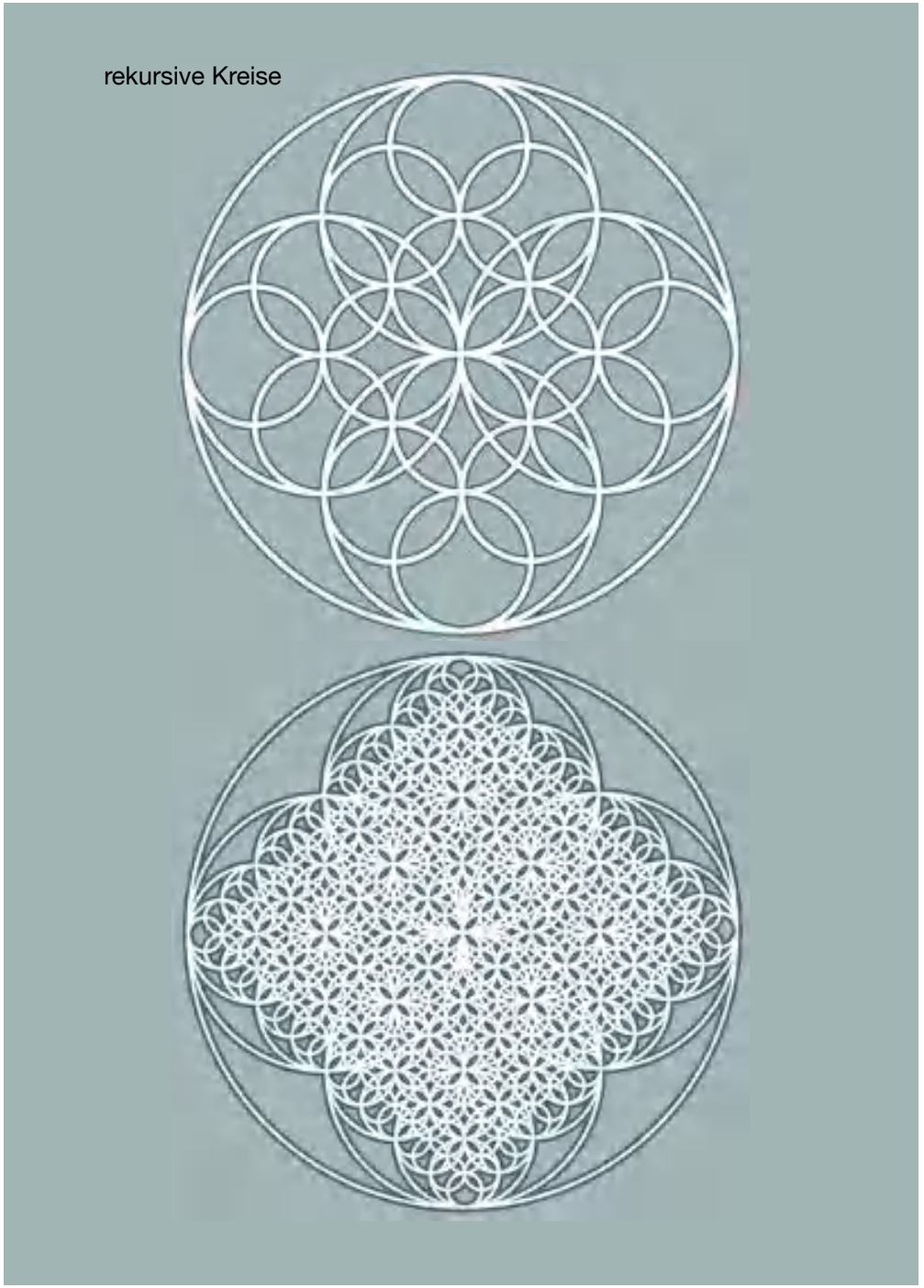


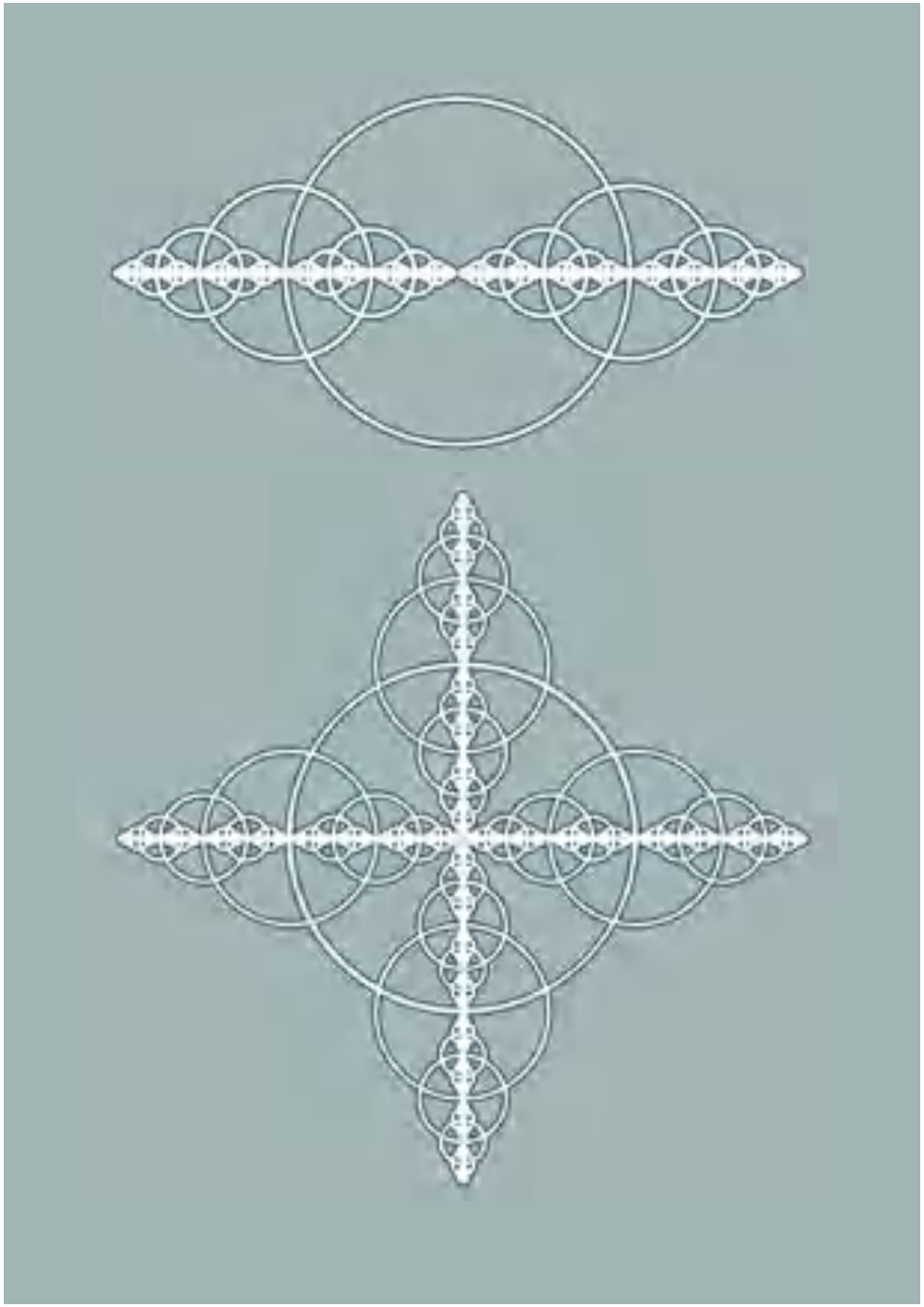
rekursive Polygonkombinationen





rekursive Kreise





Superzeichen

Das Prinzip der folgenden Beispiele ist die **Drehung** und **Spiegelung** von Linien, Kreisbögen oder Polygonen. So lassen sich neue „Superzeichen“ erstellen. Darunter verstehen wir wahrgenommene Gesamtheiten, die aus den elementaren Zeichen zusammengesetzt sind.

Dann lassen sich durch **Wiederholung** und **Kombination** der neuen Zeichen in einem Raster interessante Endlosmuster erzeugen. Solche periodischen Wiederholungen sind häufig In dekorativen Mustern zu finden.

Für die folgenden Anwendungen werden jeweils vier gleichartige Grafikelemente erzeugt für **links_unten**, **links_oben**, **rechts_unten** und **rechts_oben**:

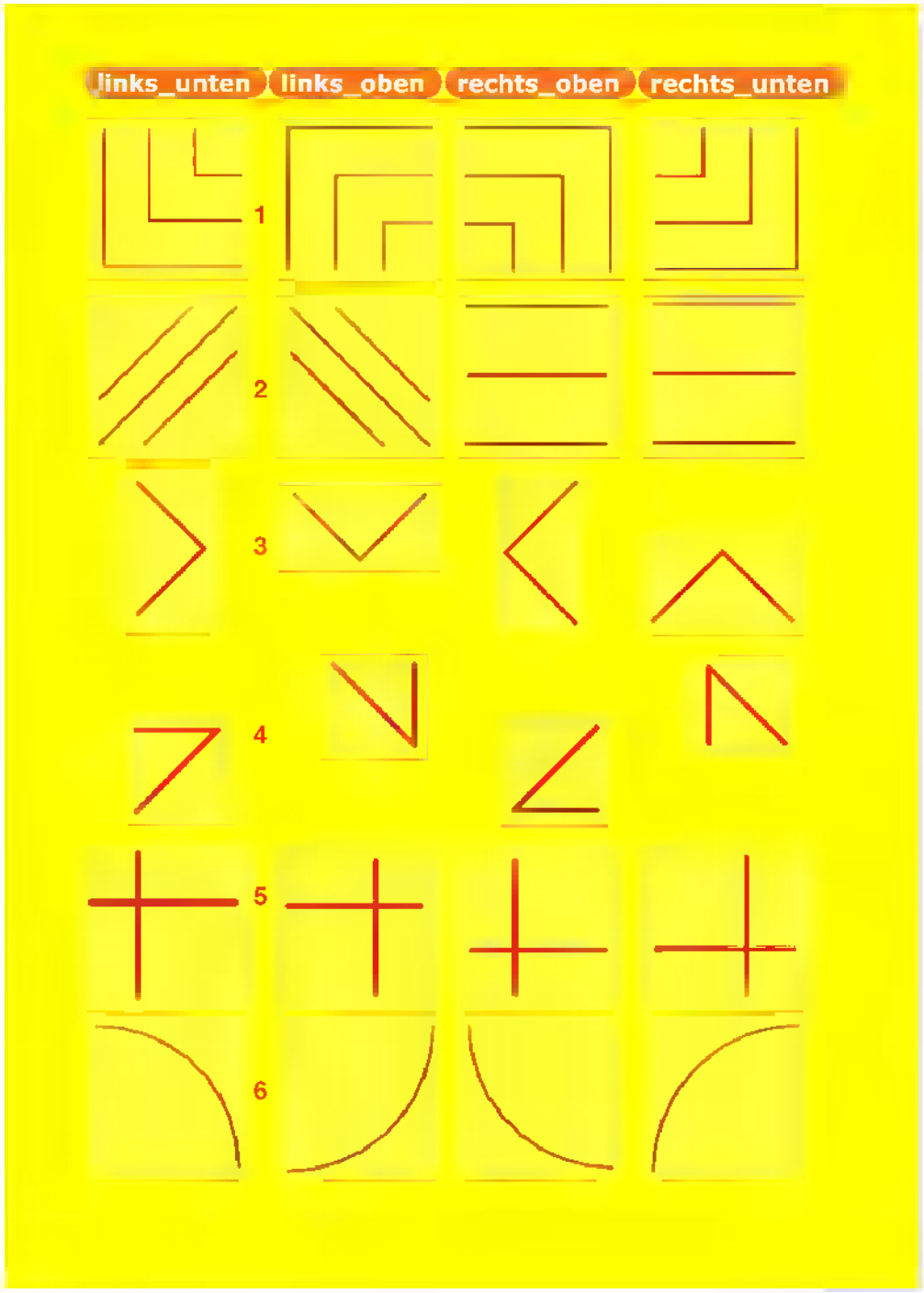
1. drei Ecken
2. (schräge) Linien
3. Dreiecke
4. Escher-Dreiecke[8]
5. Balken
6. Kreisbögen

Die Prozeduren für die Grafikelemente sind intern immer gleich aufgebaut. Das Beispiel **links_unten** zeigt (für schräge Linien), dass dafür jeweils einfache Befehle für die Bewegung der Schildkröte ausreichen.

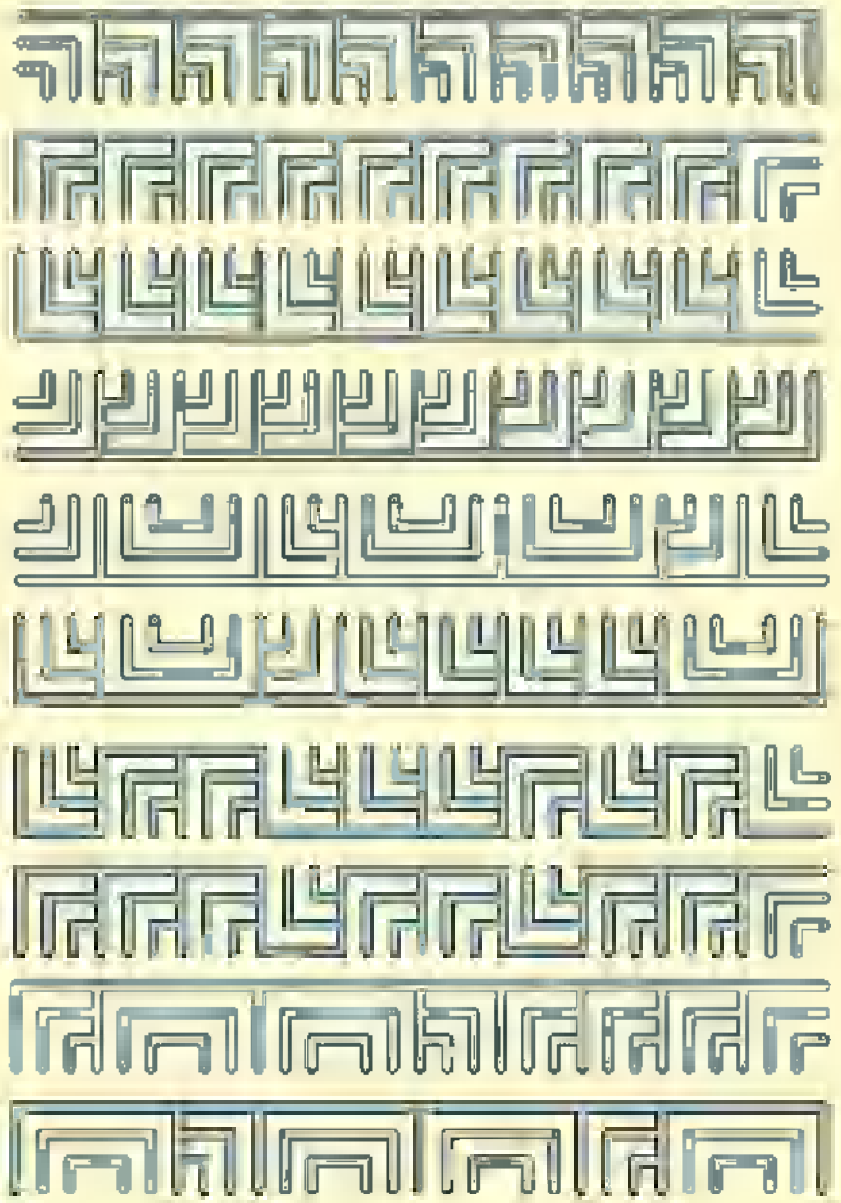
Es ist darauf zu achten, dass die Schildkröte nach dem Zeichnen wieder am Ausgangspunkt steht.

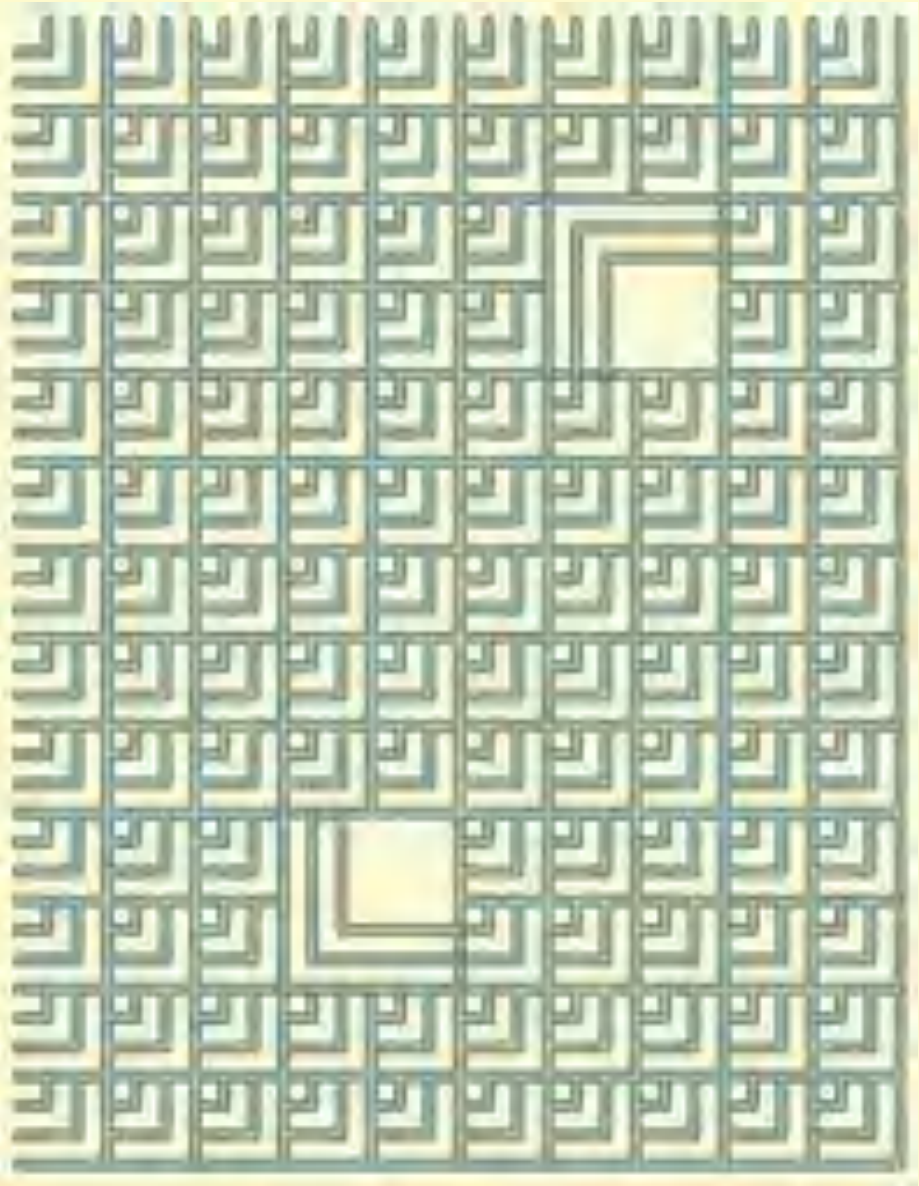
Dann kann die Wiederholung der Elemente innerhalb von Schleifen systematisch oder zufällig erfolgen.

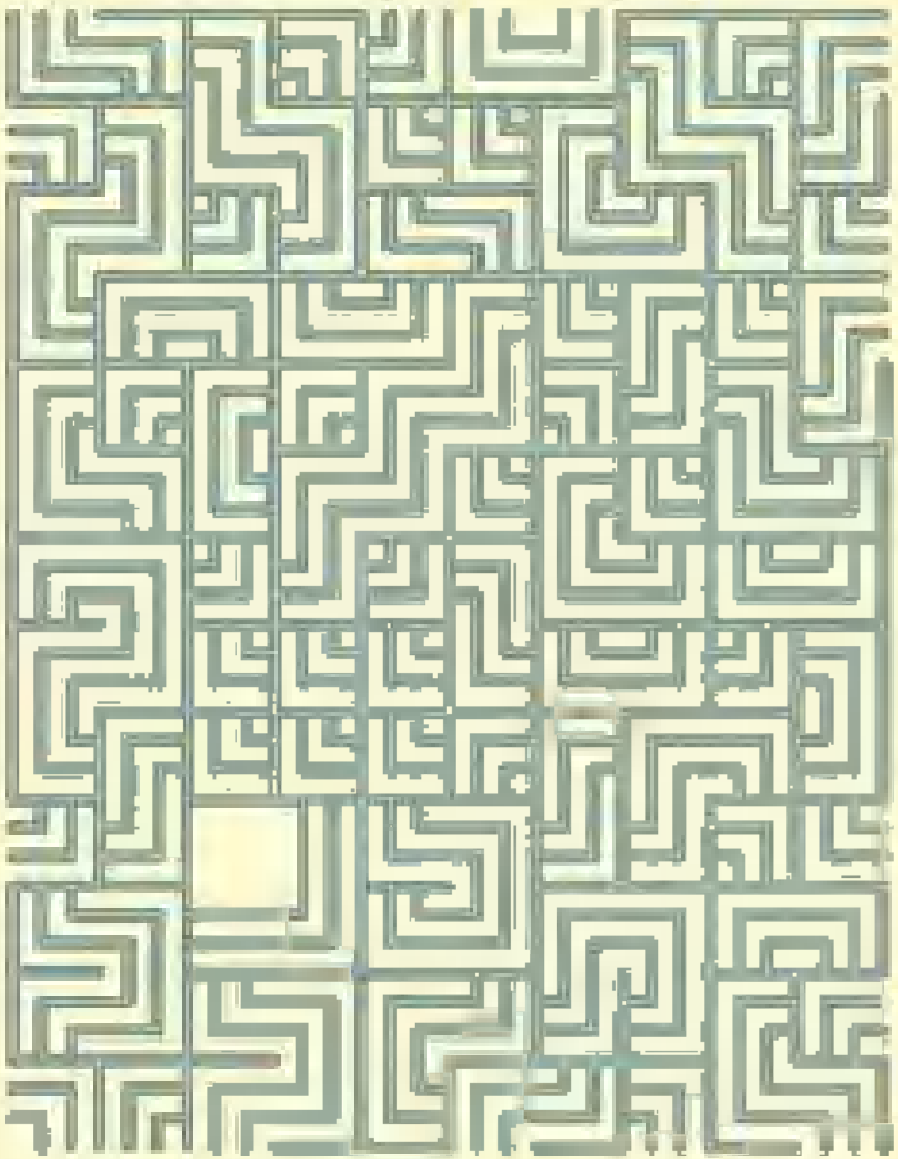
[8] Das Beispiel findet sich in einem seiner frühen Arbeitshefte, in denen er mit solchen Mustern experimentierte (Schattschneider, 1990, S. 45).



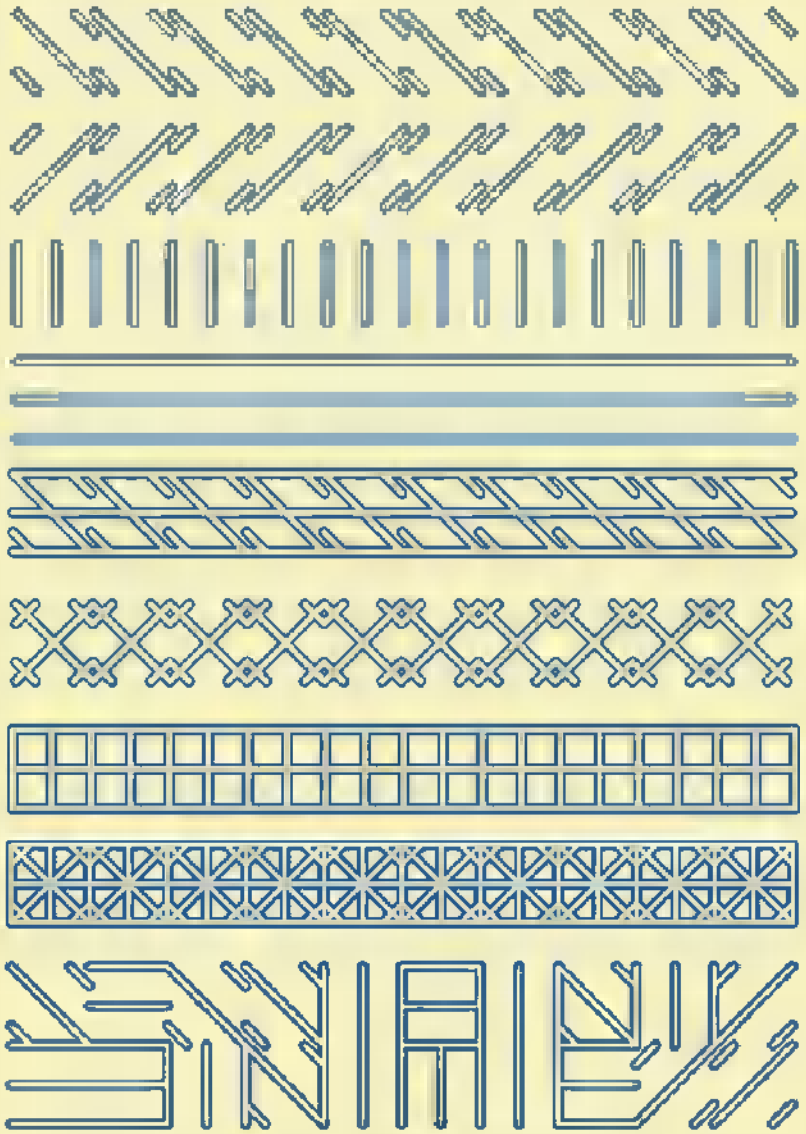
drei Ecken

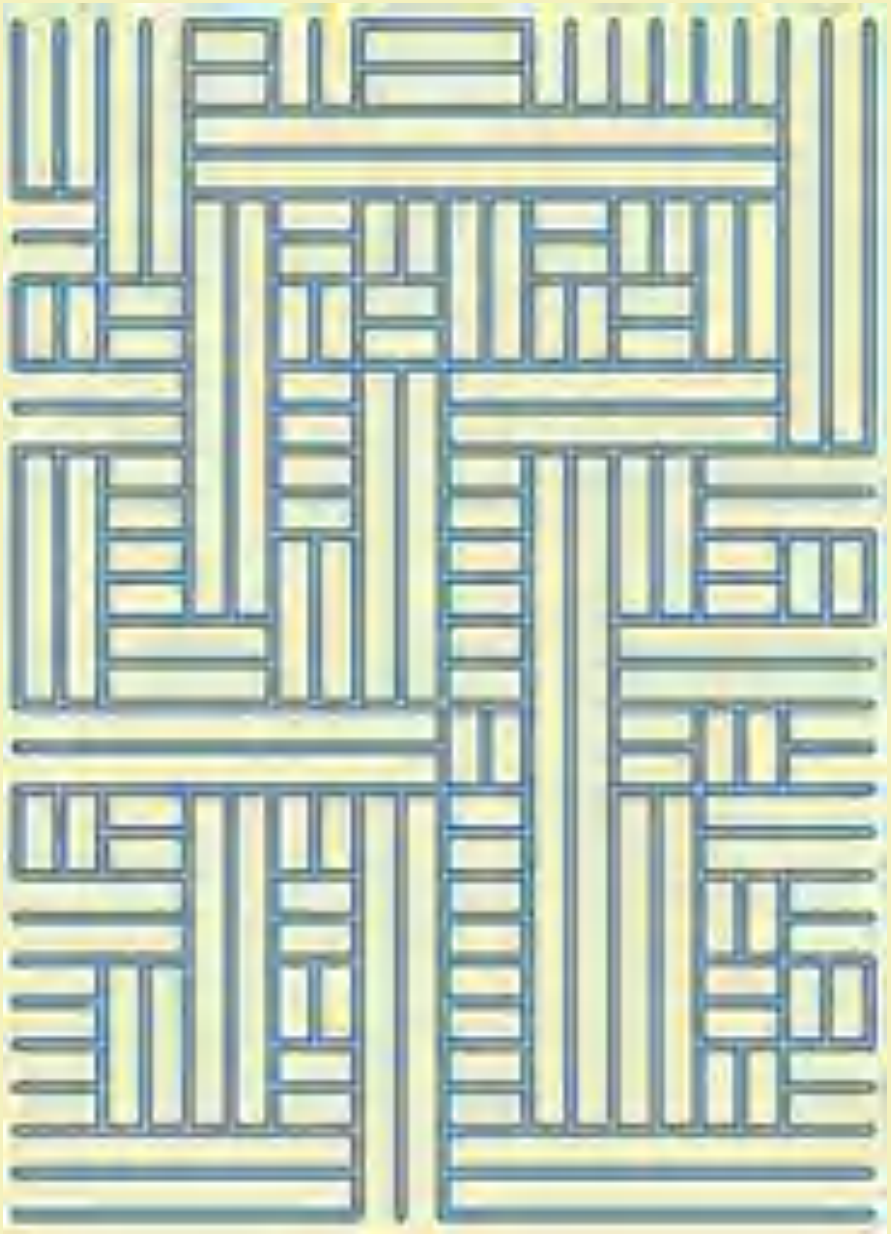


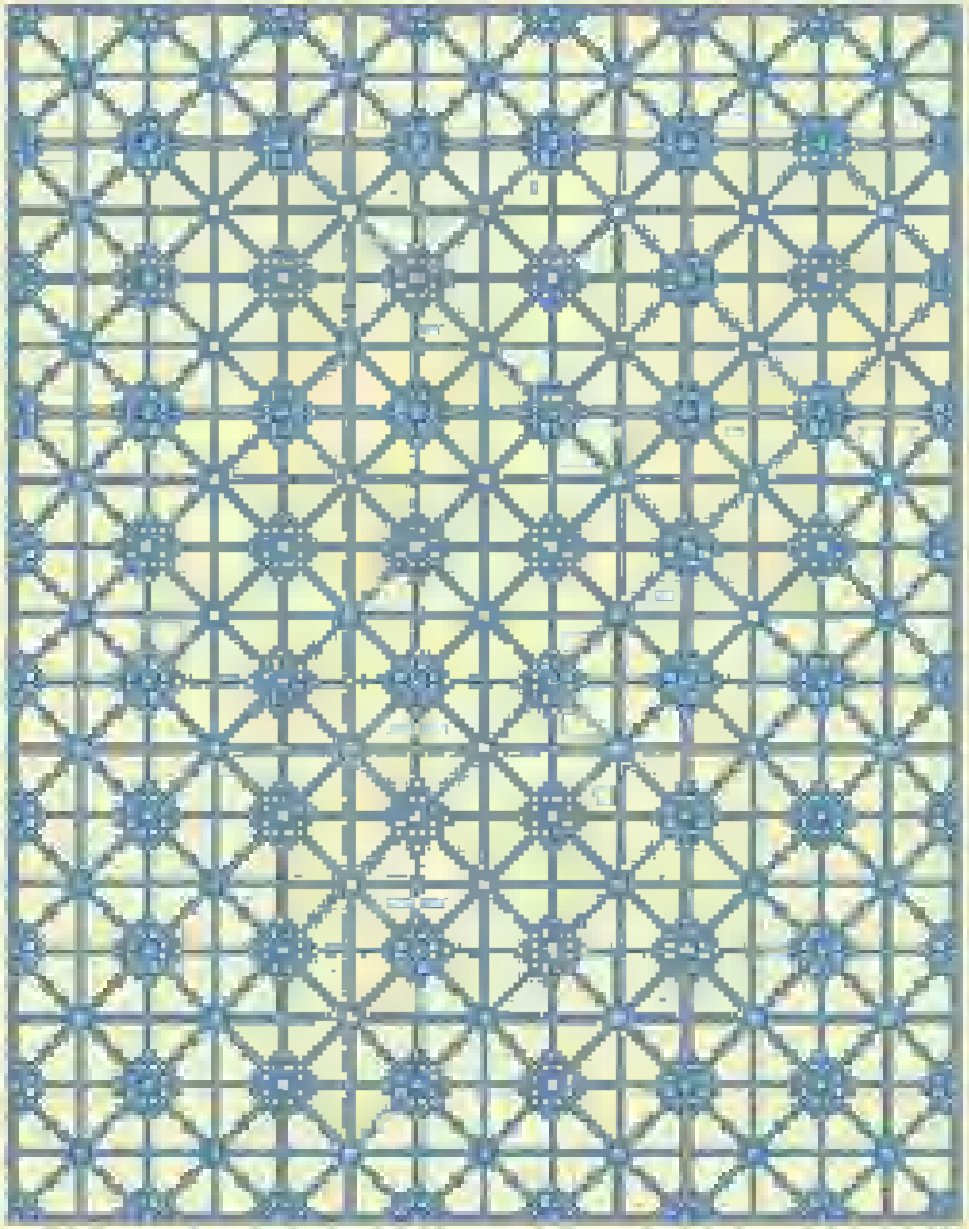




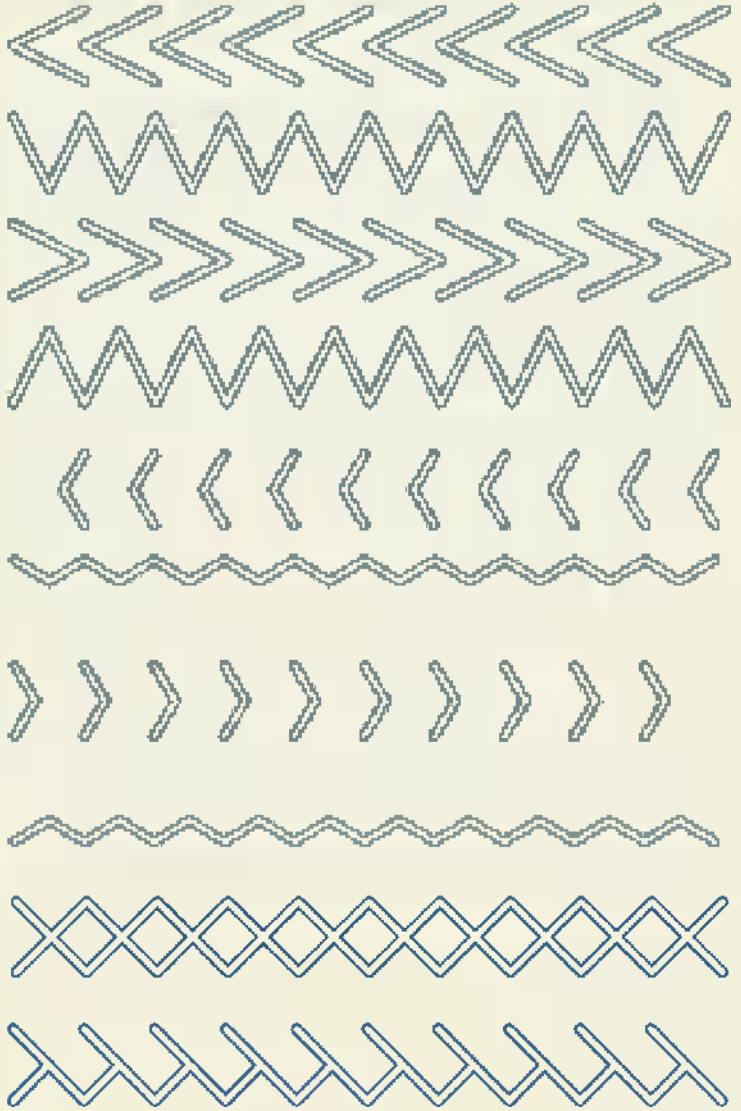
schräge Linien

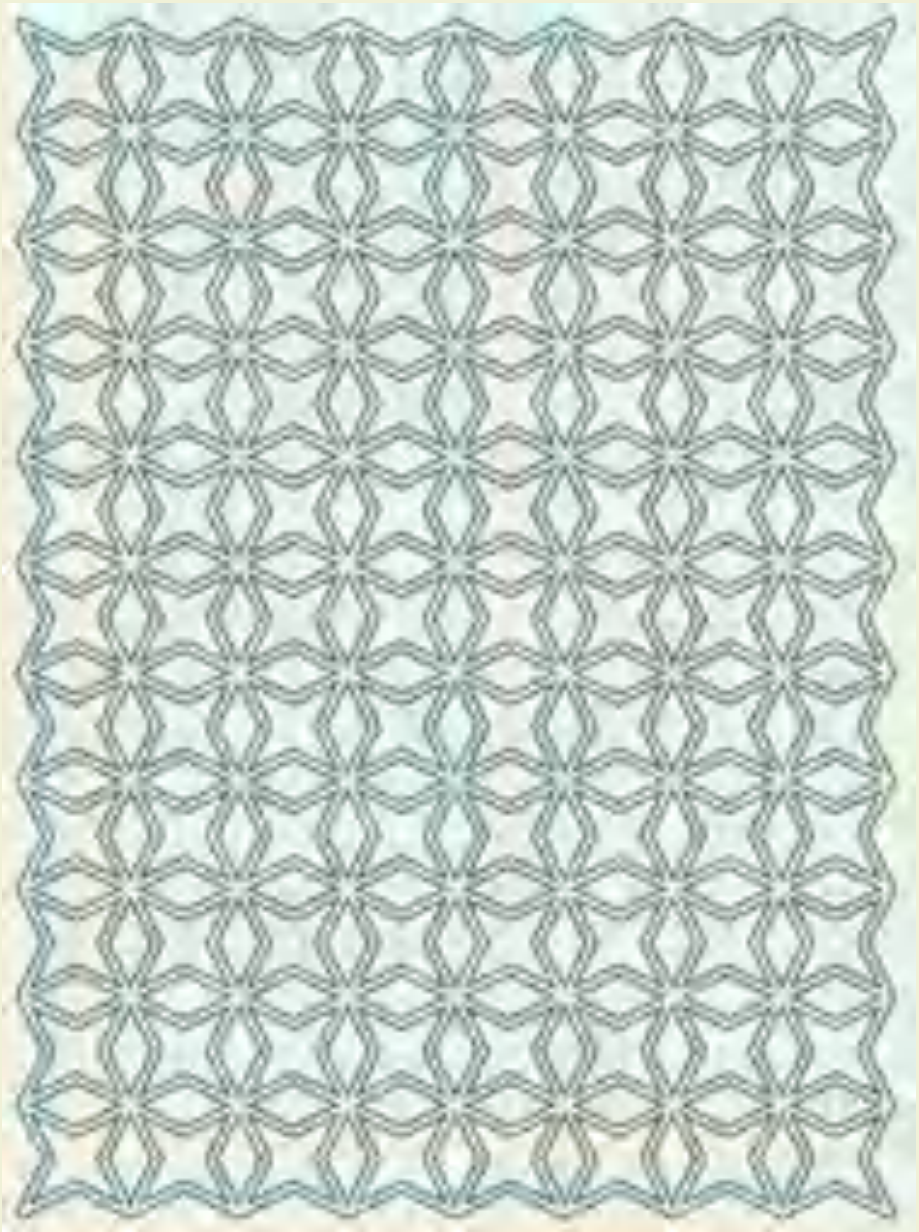


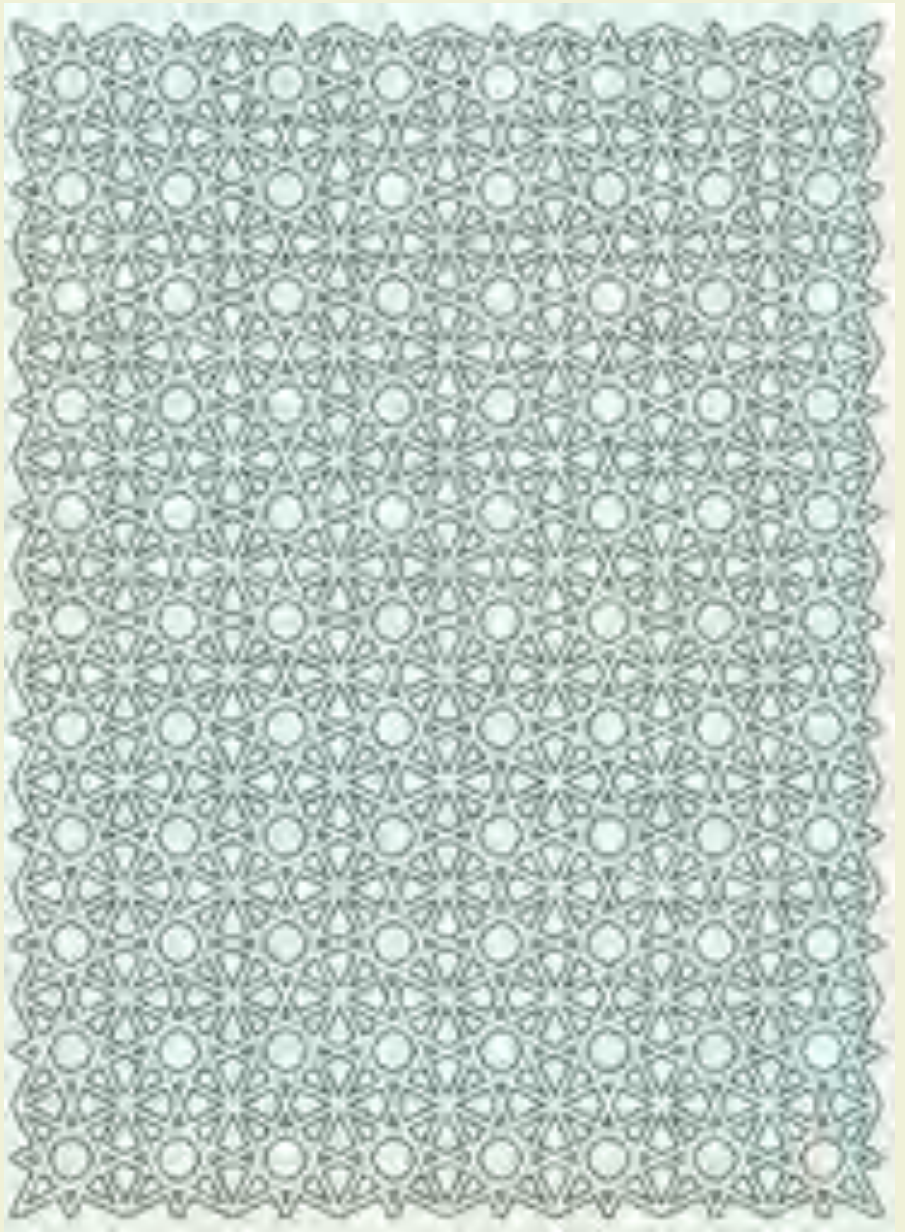


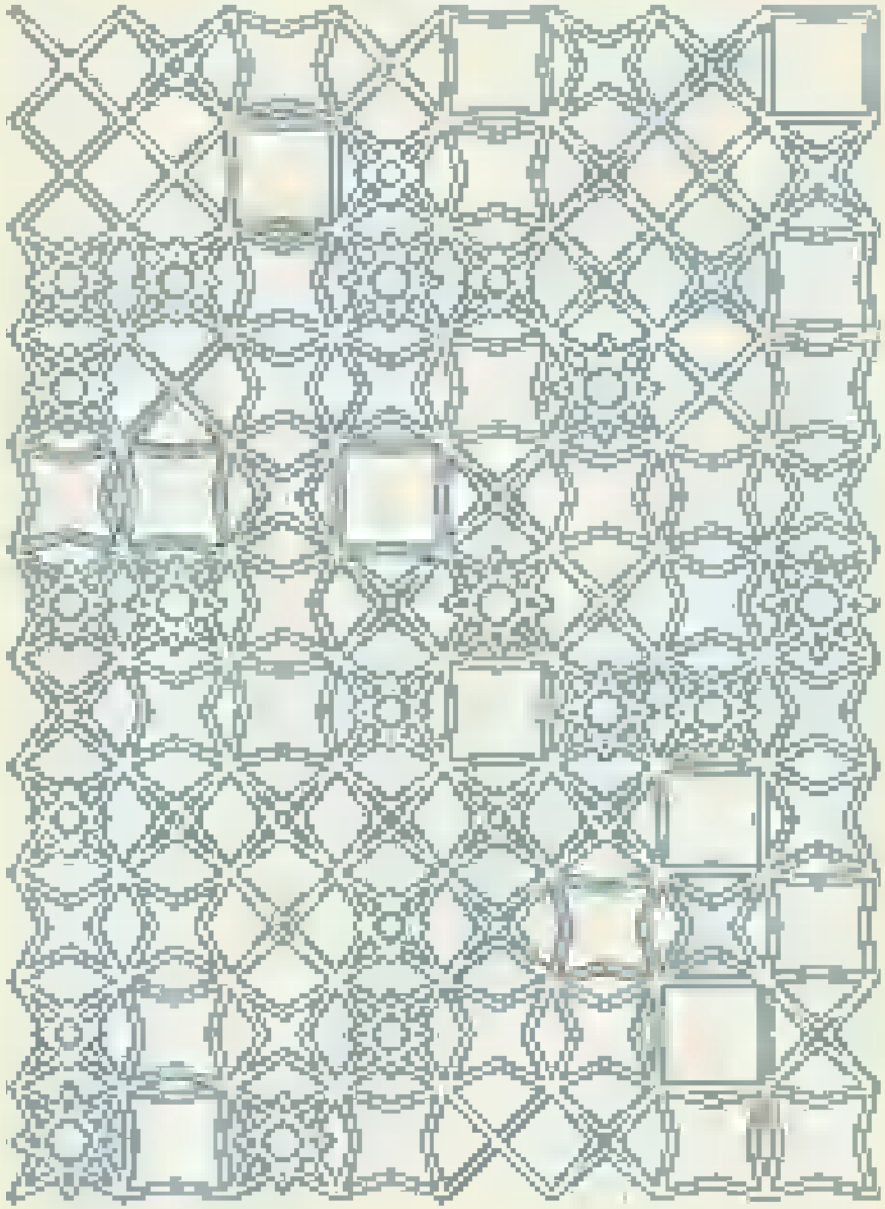


Dreiecke

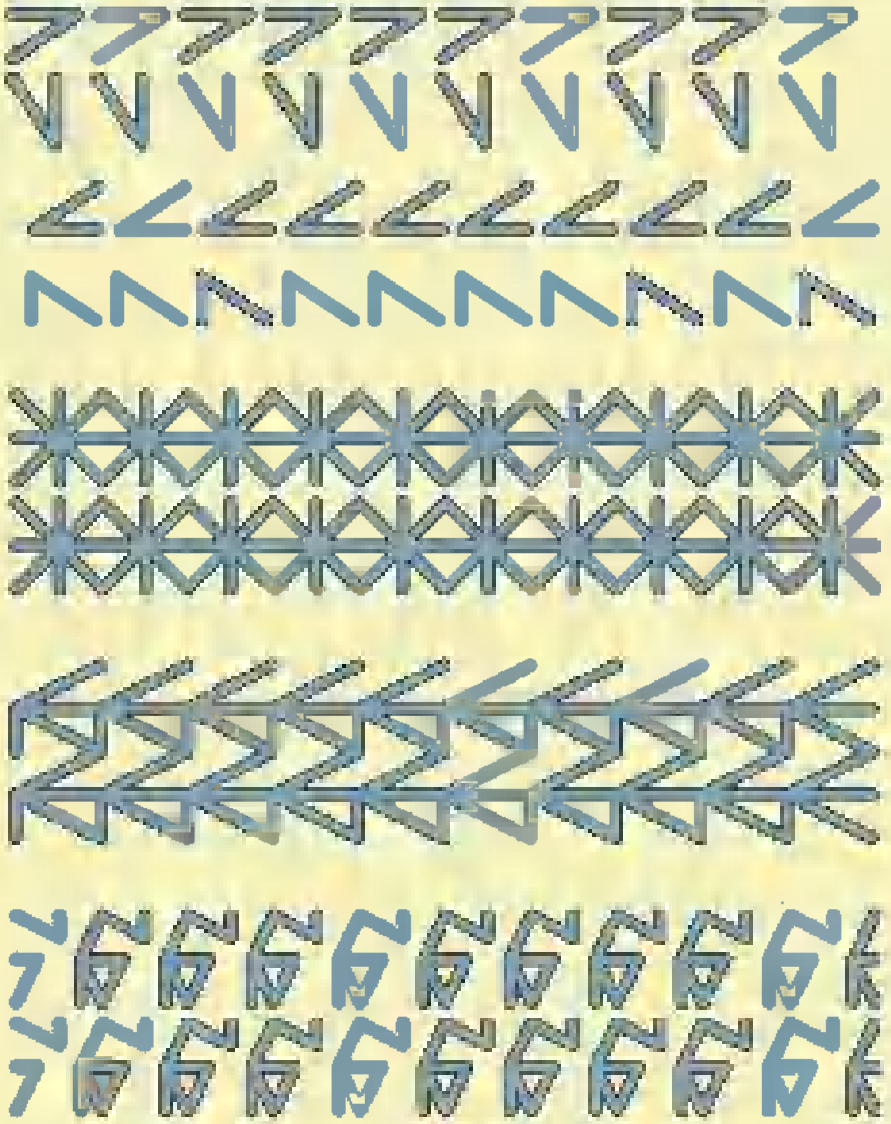


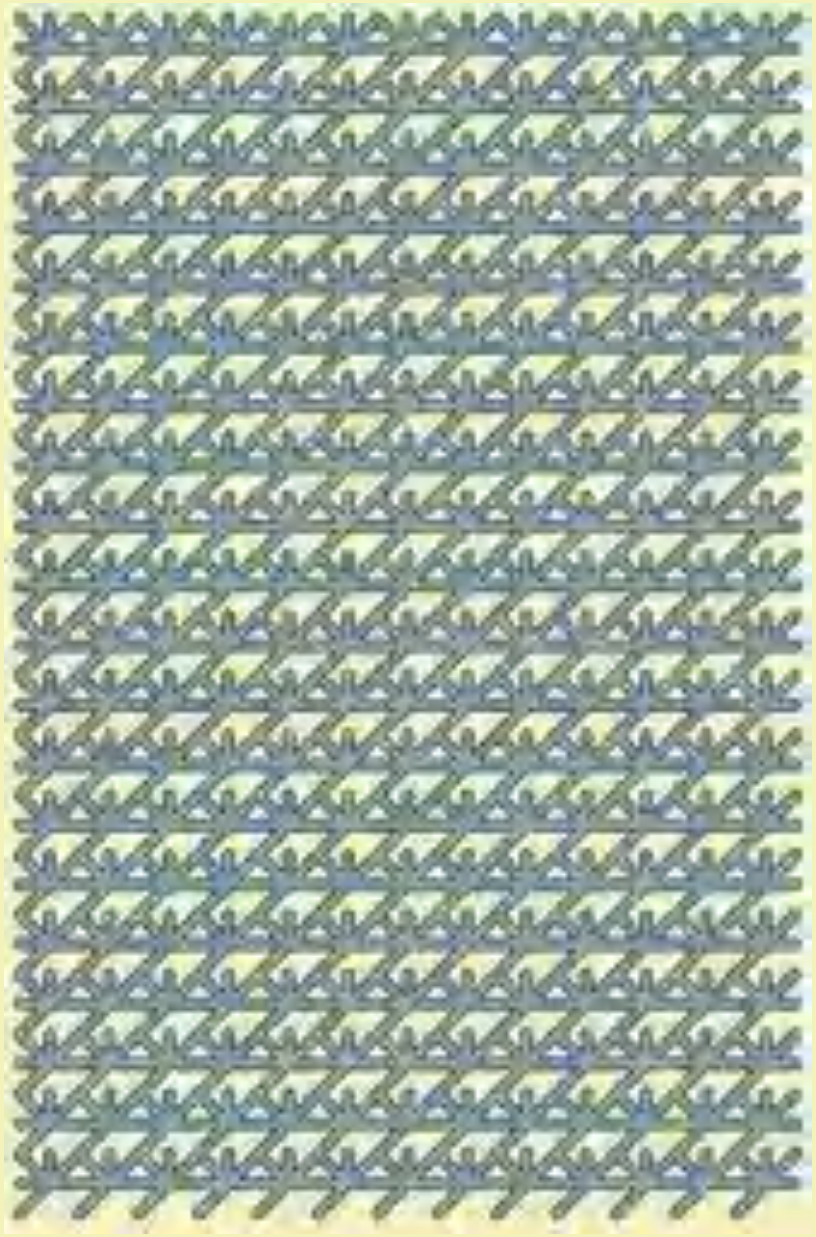


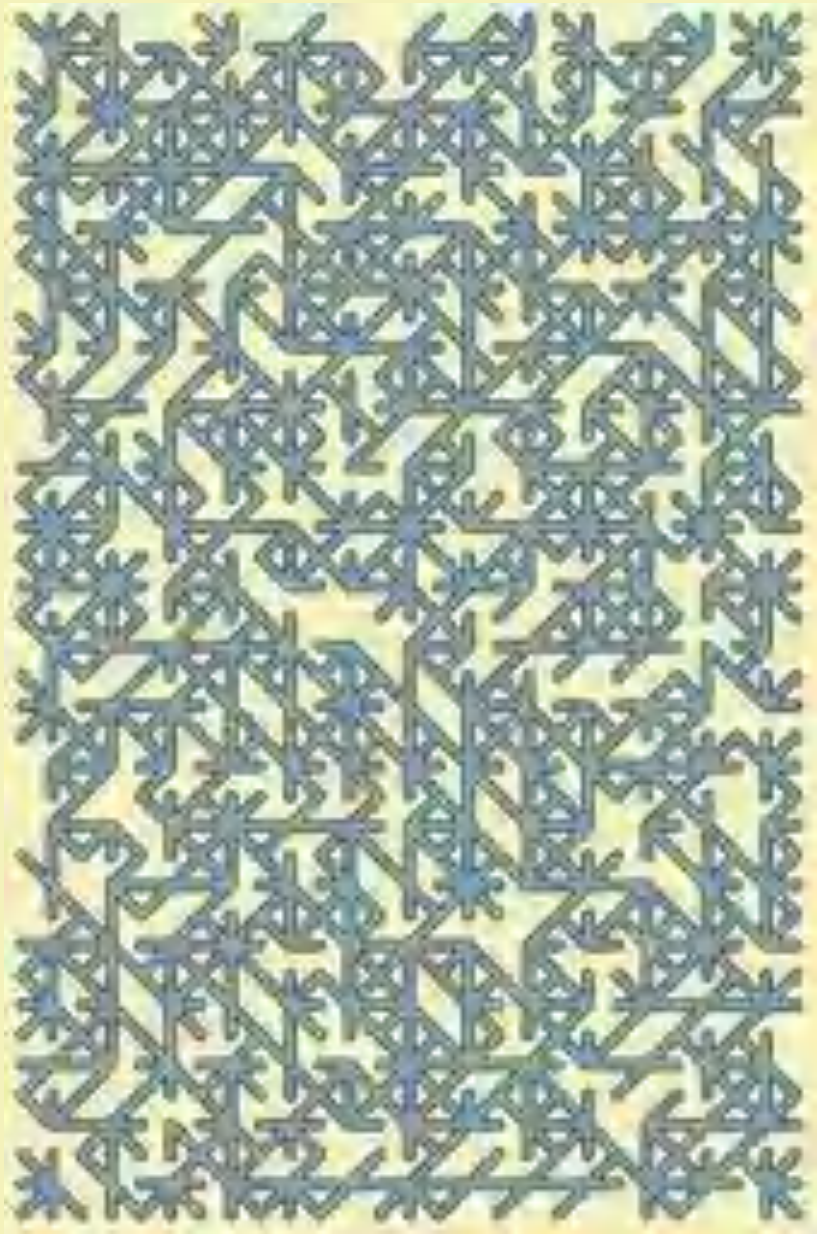


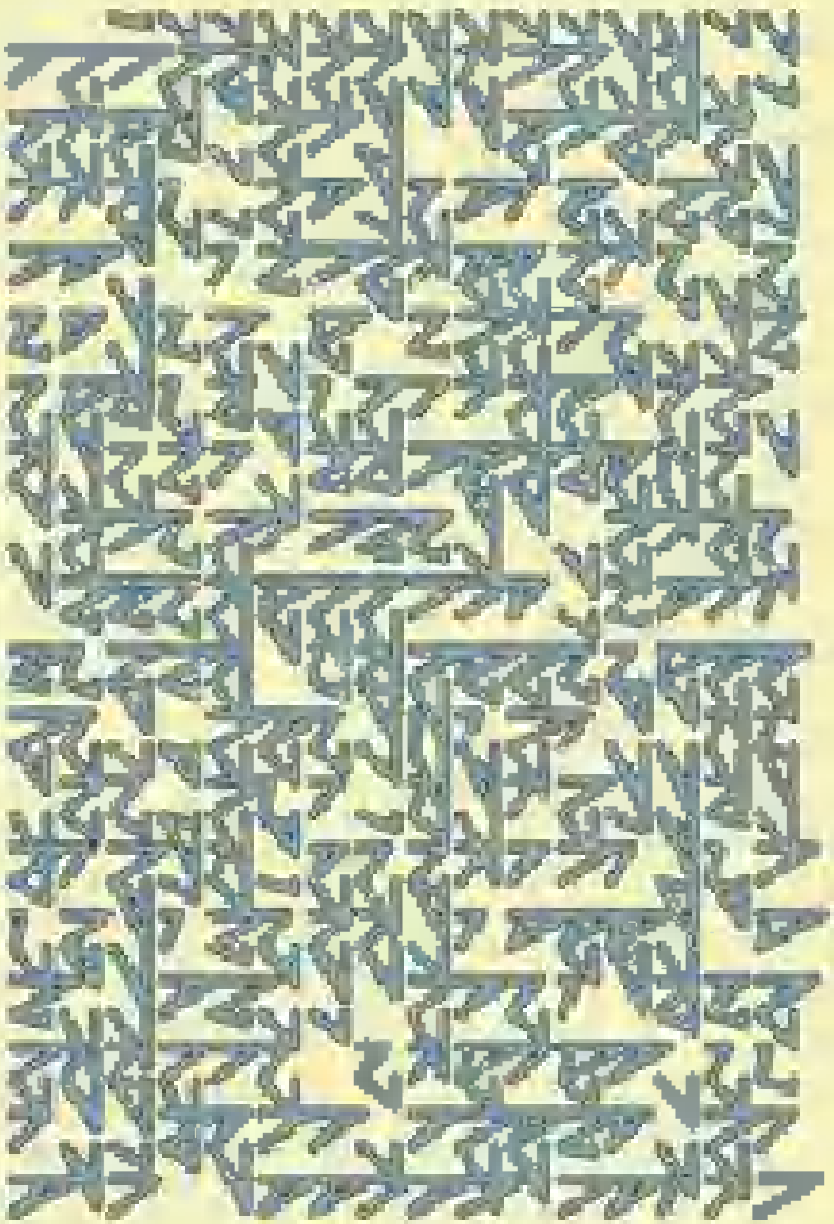


Escher-Dreiecke

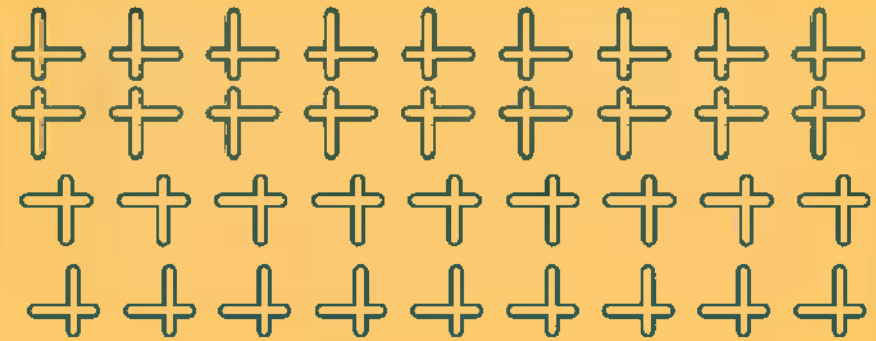


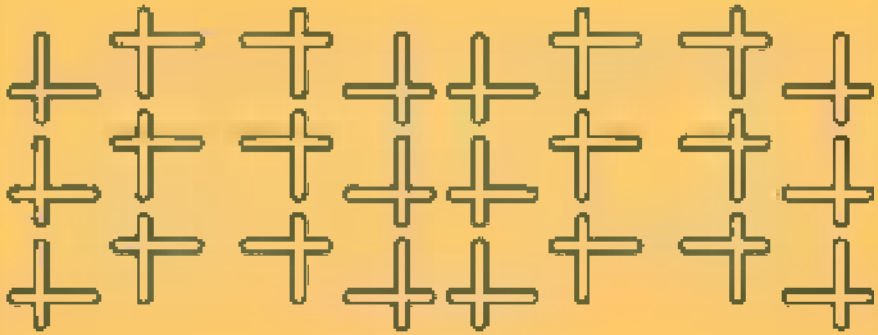


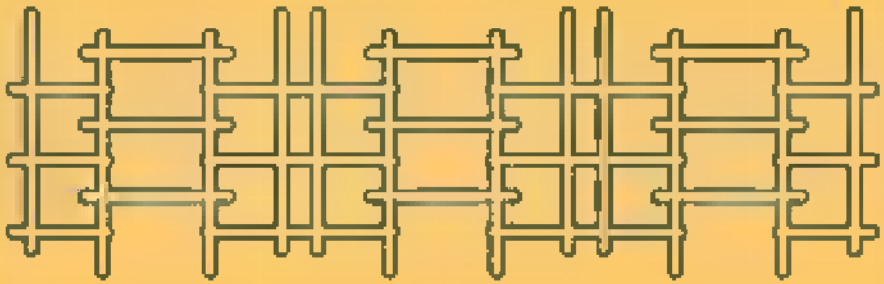


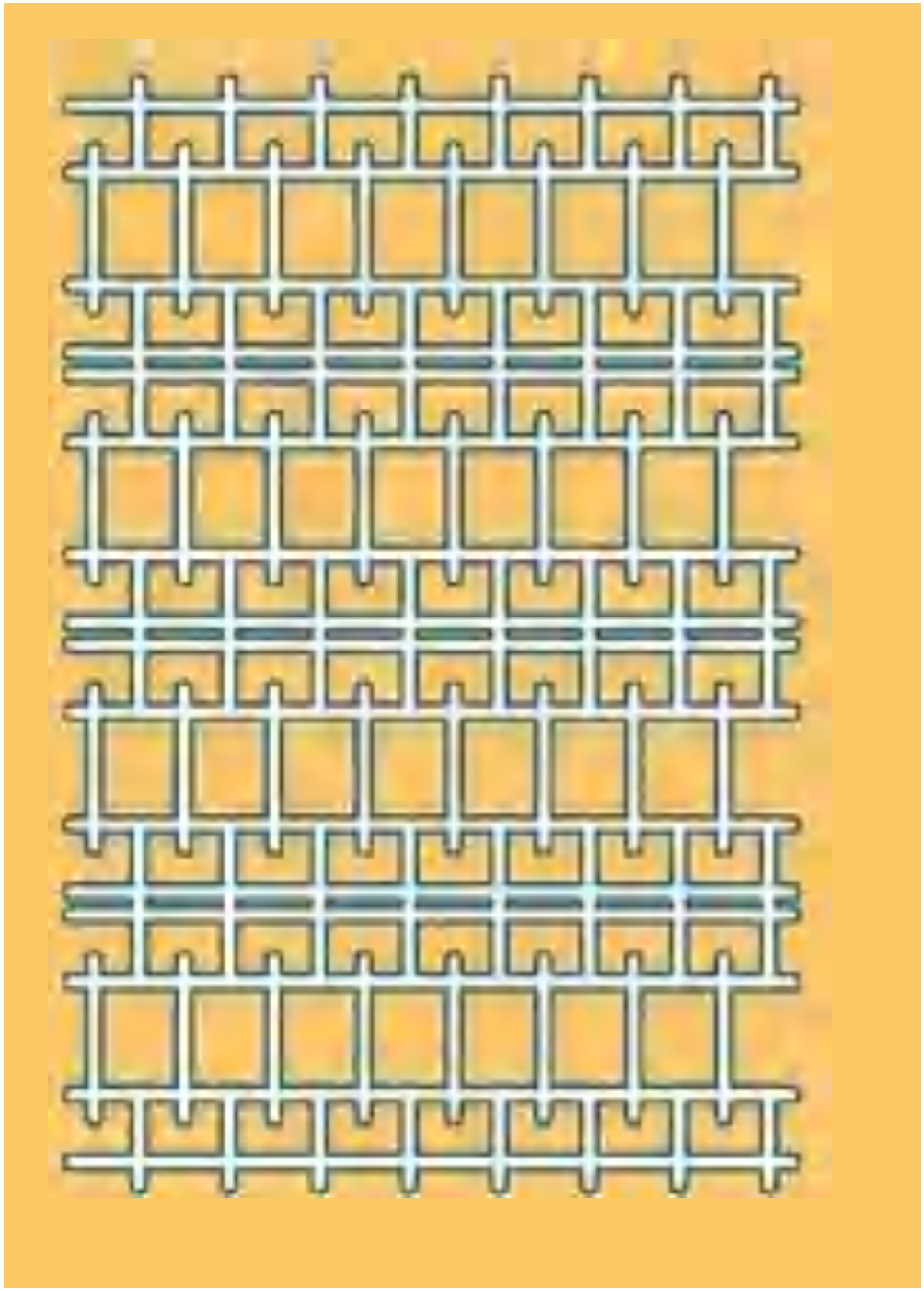


Balken

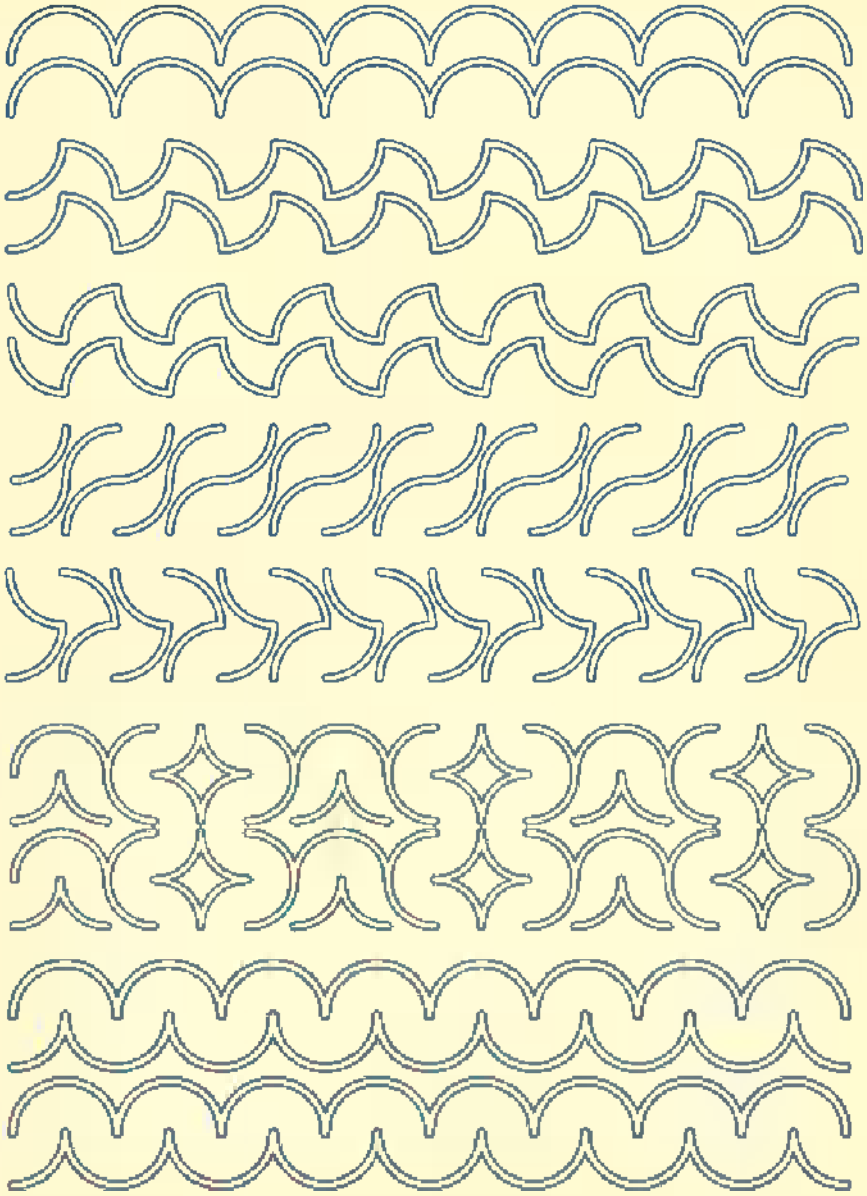


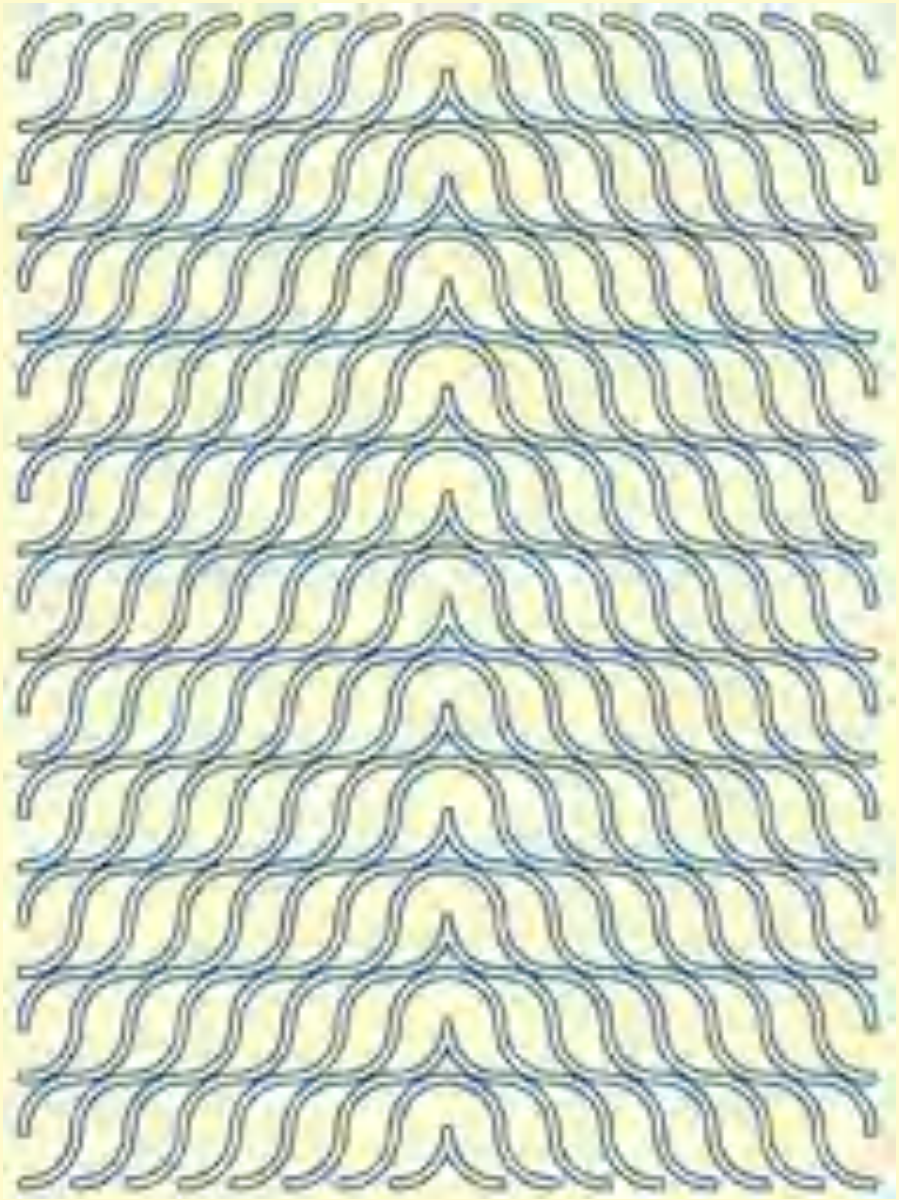


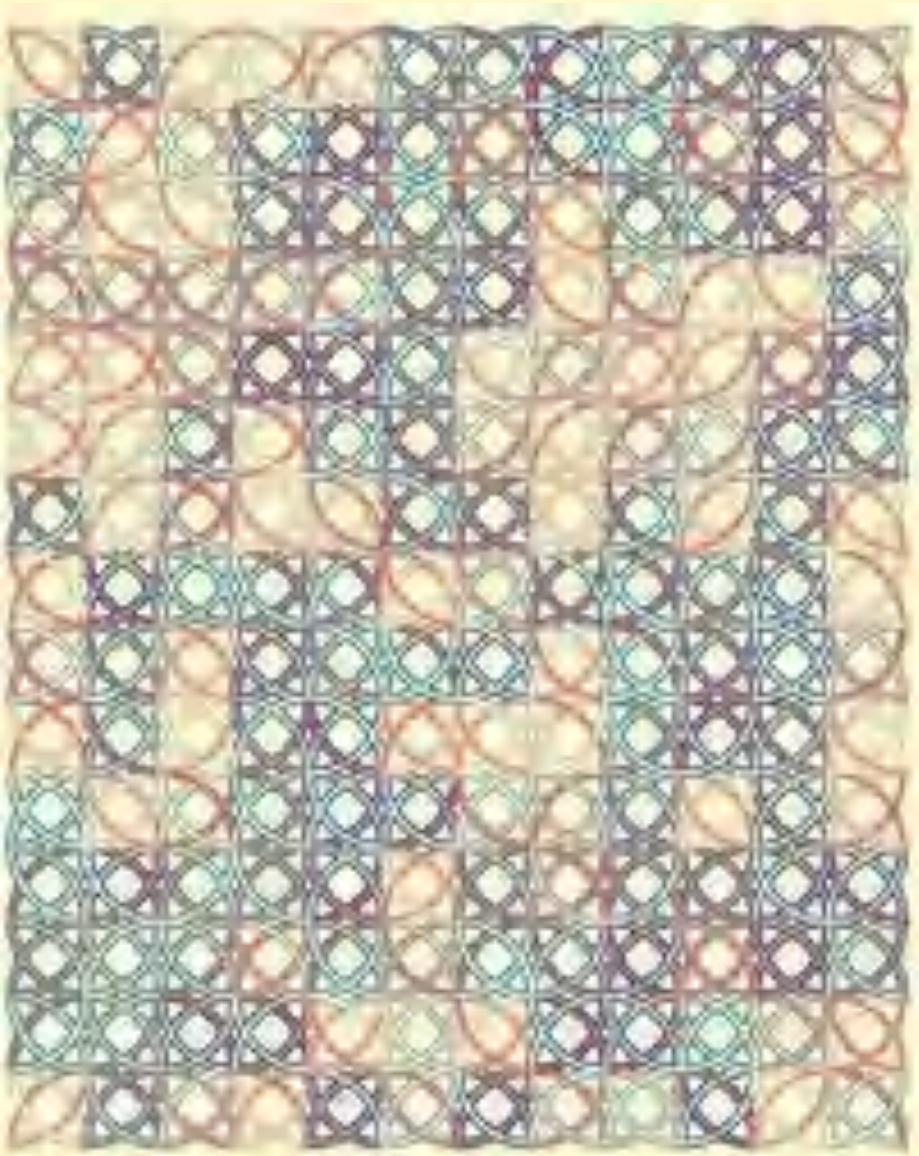


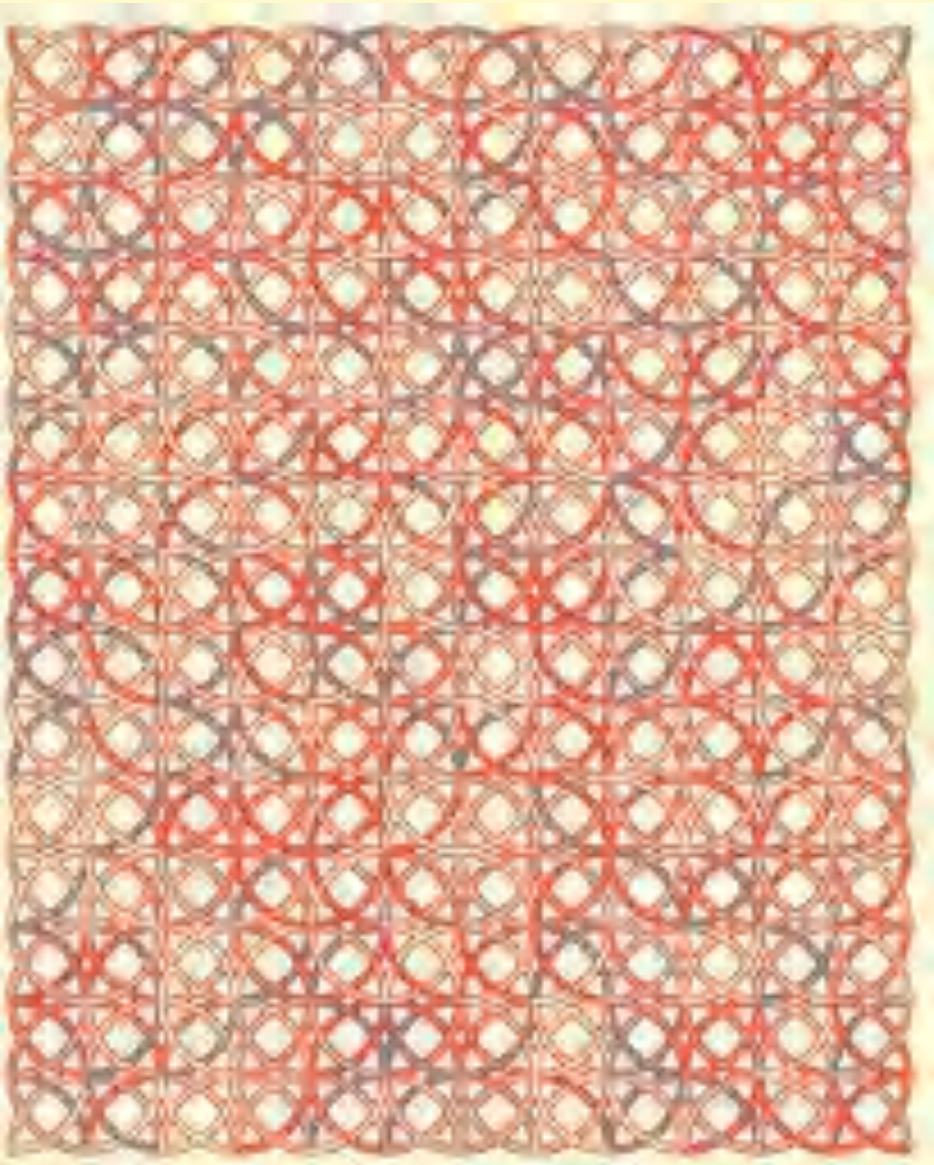


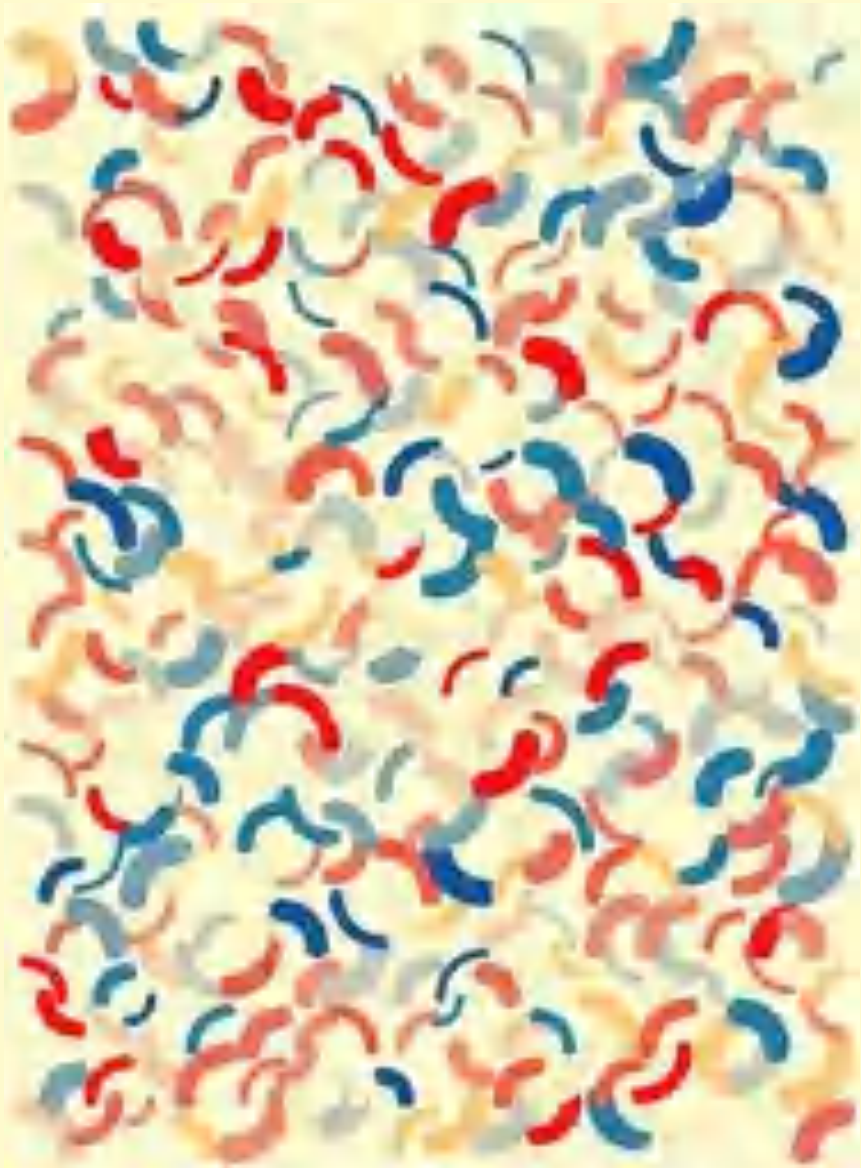
Kreisbögen











Überlagerungen (Moiré)

Mit **Überlagerungen** der nun schon bekannten grafischen Elemente (Linien, Kreise, Punkte) lassen sich überraschende optische Wirkungen erzielen (es wird dann vom Moiré-Effekt gesprochen). Ihre Erzeugung ist denkbar einfach[9].

Das Vorgehen ist durchgängig dasselbe (der blaue Punkt in den Grafiken rechts markiert immer den Bildmittelpunkt):

- Es wird ein erstes Grundmuster mit einem Element aus dem Figurenbaukasten erzeugt. Dieses Muster bildet den Hintergrund (im Bild rechts also von oben nach unten Linien, Kreise, Punkte und wieder Linien).
- Je nach angestrebtem Effekt können eines oder mehrere weitere Muster erzeugt und dem ersten Muster überlagert werden.
- Das zweite Muster ist gegenüber dem ersten meist leicht versetzt und/oder verdreht zu zeichnen (im Bild rechts in der Mitte von oben nach unten leicht verschobene Linien, Kreise, Punkte und wieder Kreise).

Die Ergebnisse finden sich im Bild rechts in der dritten Spalte ganz außen.

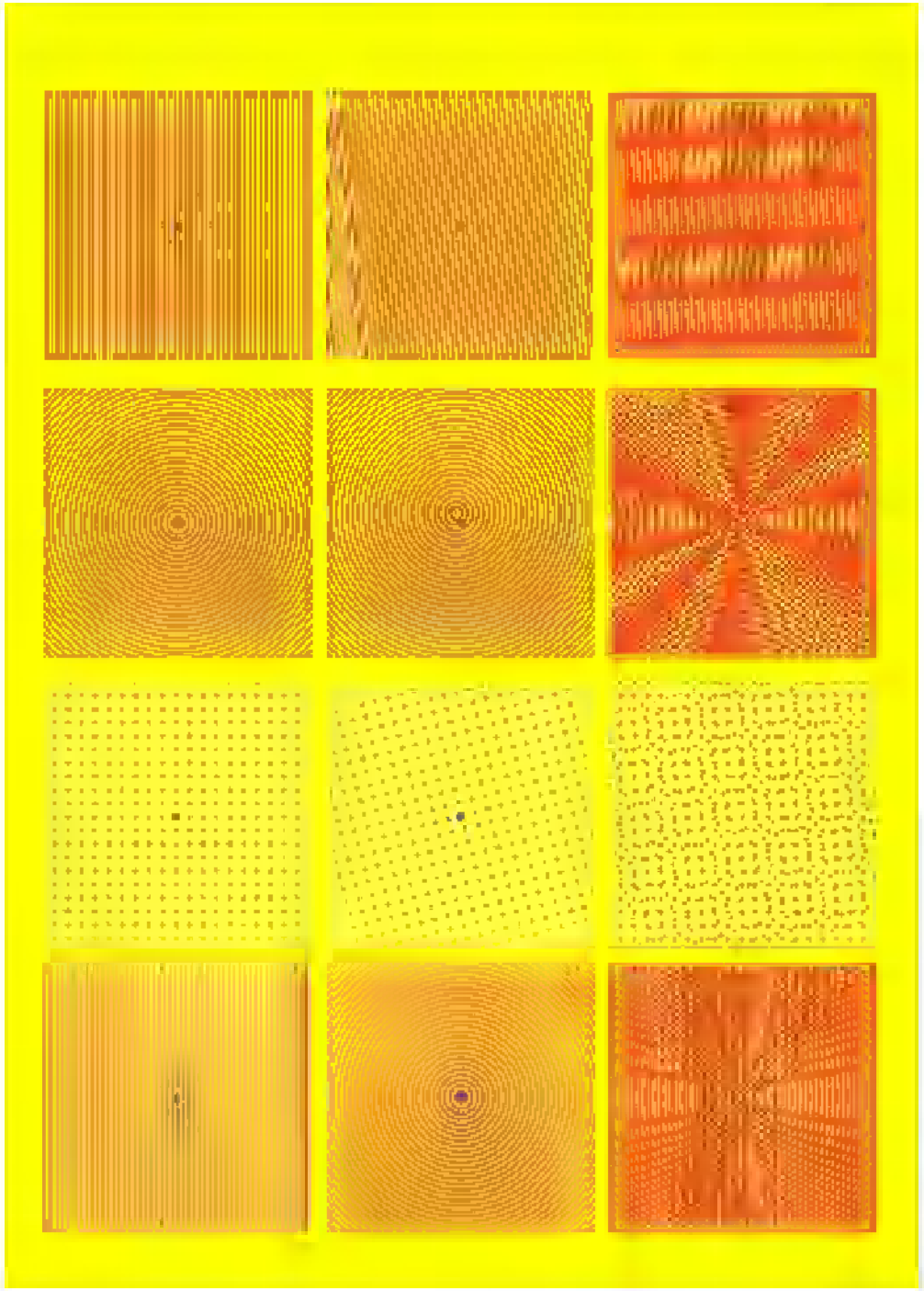
Moiré-Muster sind in verschiedenen technischen Bereichen von Bedeutung. Sie werden aber auch seit der Op-Art bildnerisch eingesetzt.

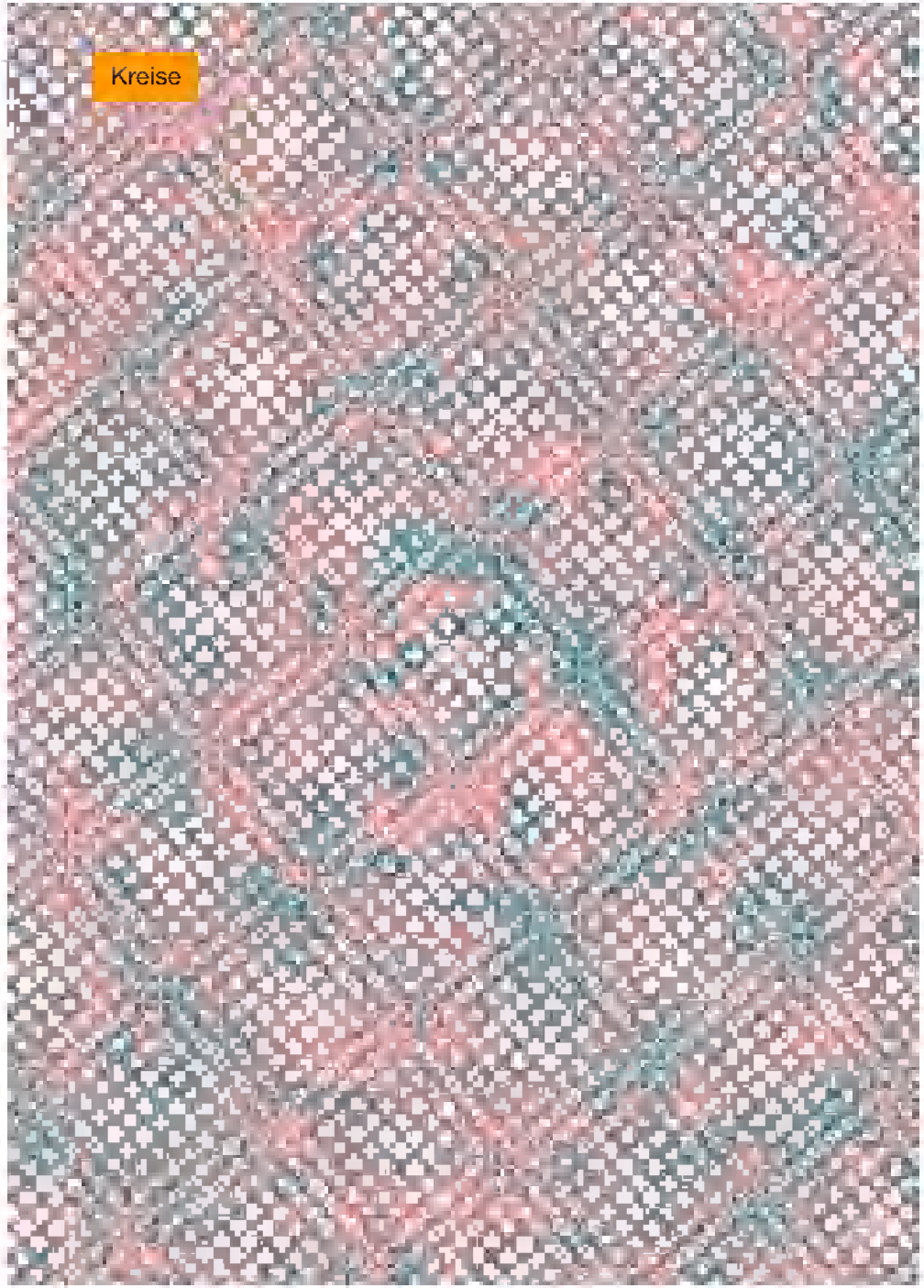
Die Kombination unterschiedlicher Formen in verschiedenen Farben und Größen liefert häufig überraschende Ergebnisse[10].

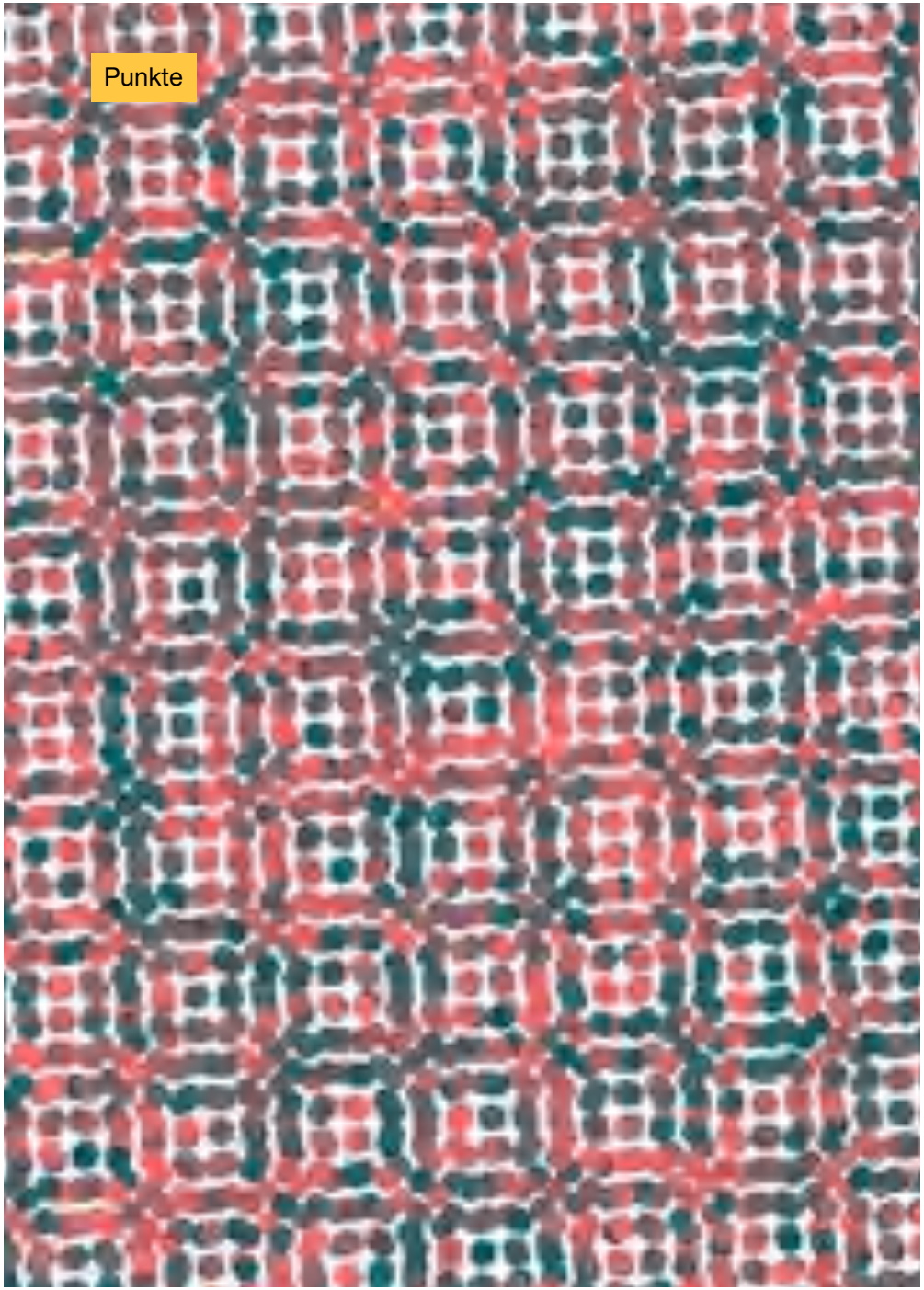
---

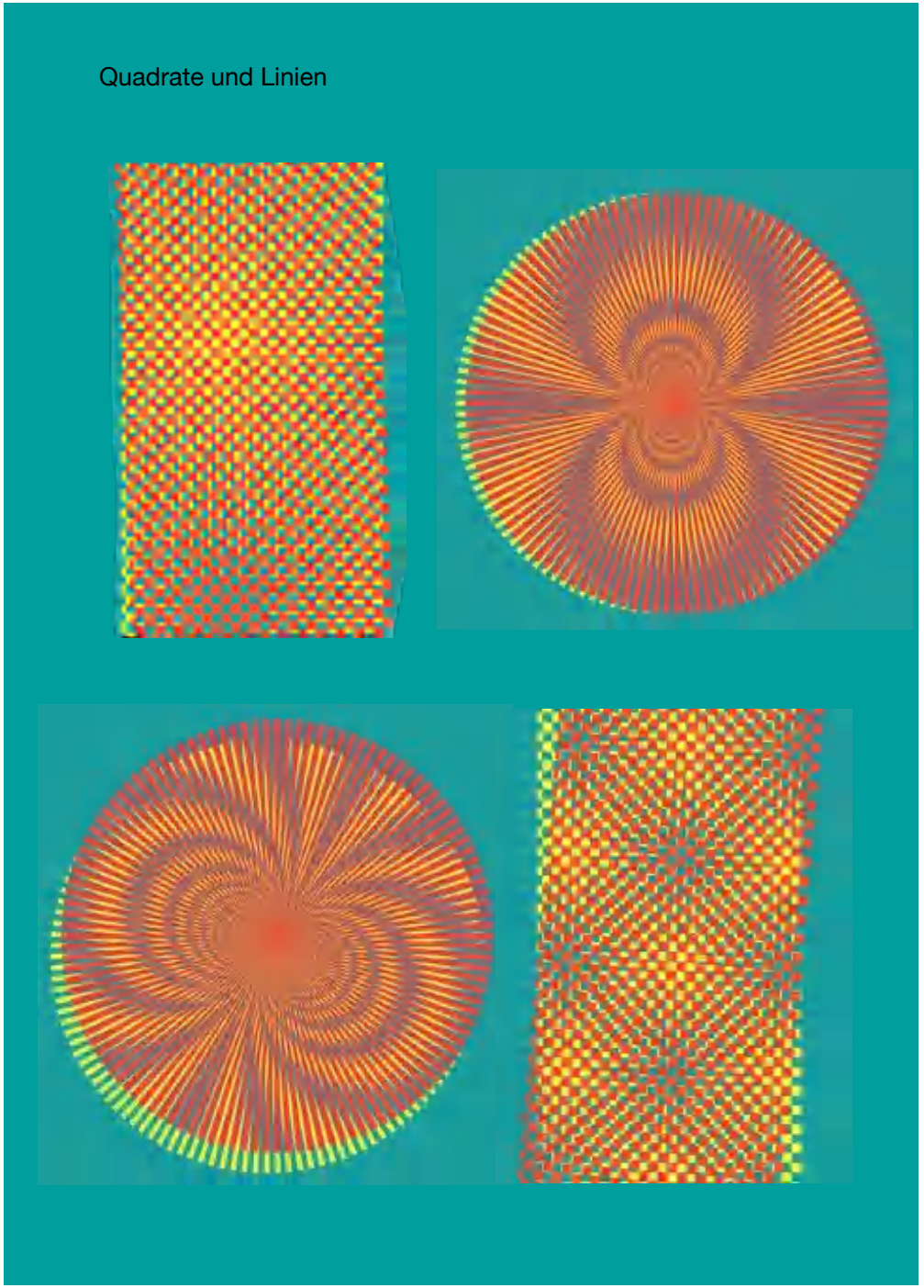
[9] Für die Darstellung eignen sich bereits statische Bilder; besonderen Reiz gewinnen sie jedoch für die Betrachter, wenn sie als programmgesteuerte Animation ablaufen oder im Idealfall selbst interaktiv (durch händische Verschiebung der Muster) beeinflusst werden können.

[10] Mit dem moiré index hat Carsten Nicolai ein ganzes Buch vorgelegt, in dem er genau solche systematischen Transformationen von Grundelementen in Gittern vornimmt. Das kann so ziemlich alles nach dem hier vorgestellten Prinzip recoded werden.

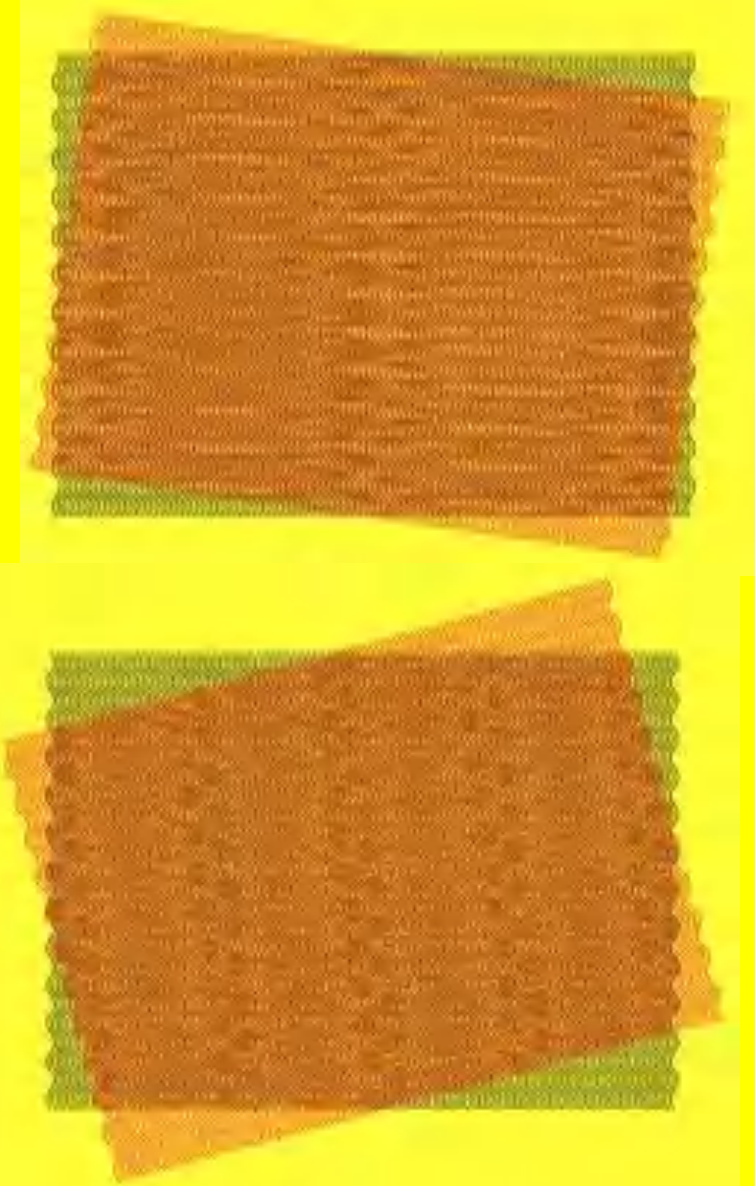


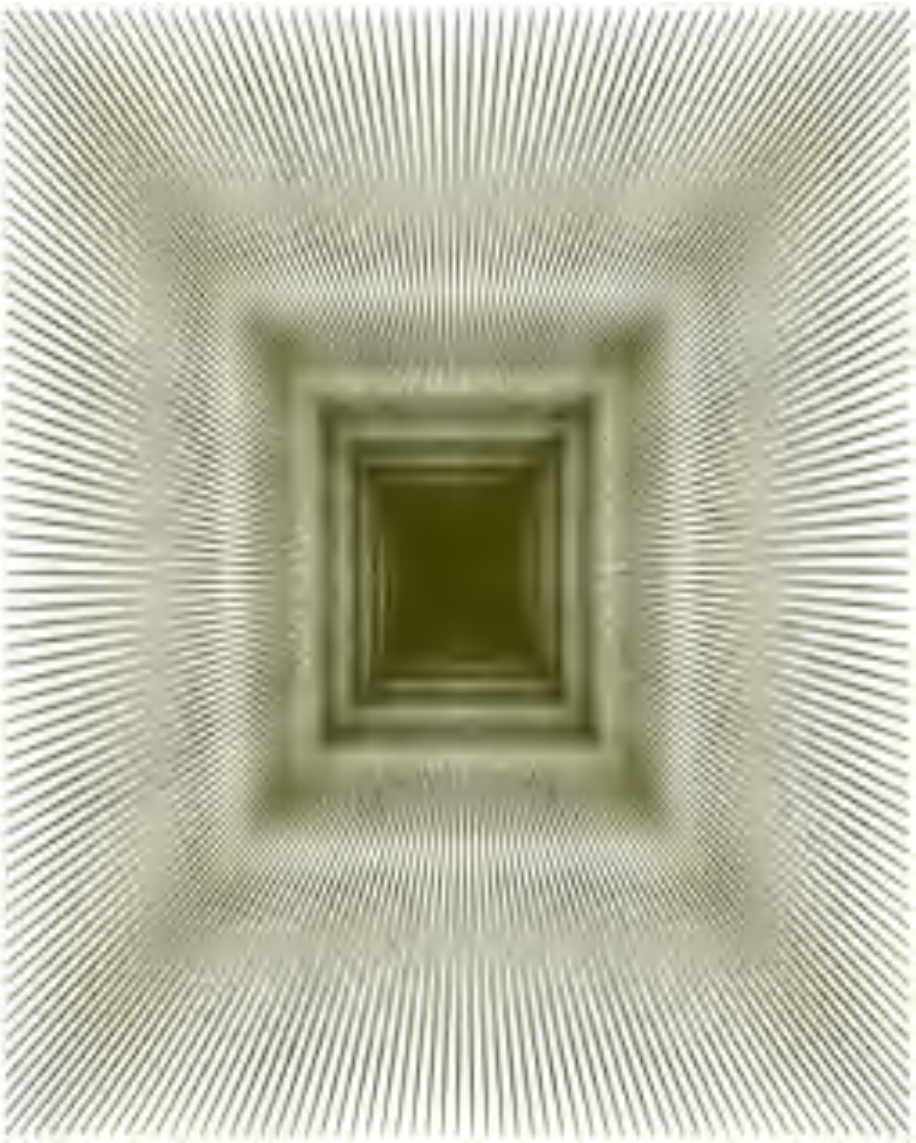

Kreise


Punkte

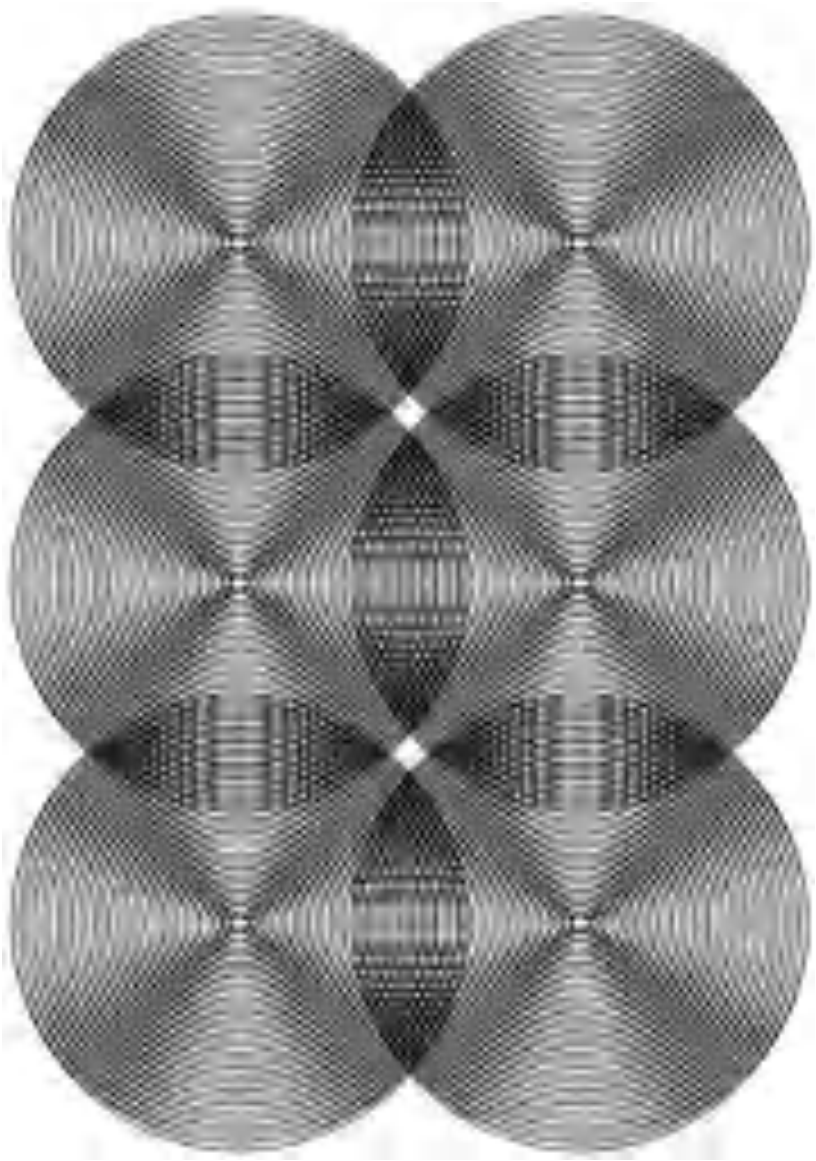

Quadrate und Linien

Kreisbögen





Hommage à Biasi: Lot Nr. 639

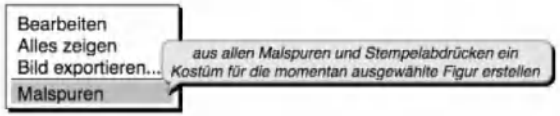


Hommage à Oxenaar: 45 Cent Stamp

Stempeln statt Zeichnen

Es gibt in der Programmierumgebung Snap! nützliche Funktionen, die das Zeichnen aufwändiger Grafiken sehr vereinfachen können. So wurden in allen bisherigen Projekten die grafischen Elemente durch die Bewegung der Schildkröte bei abgesenktem Stift erzeugt. Es ist aber auch möglich, die Schildkröte - in Form ihres **Kostüms** - selbst zu einem grafischen Element zu machen.

Für jede angefertigte Grafik kann durch **Rechtsklick** auf der **Bühne** ein Kontextmenü geöffnet werden, in dem durch die Wahl der Option **Malspuren** aus der Grafik ein solches Kostüm erzeugt wird.

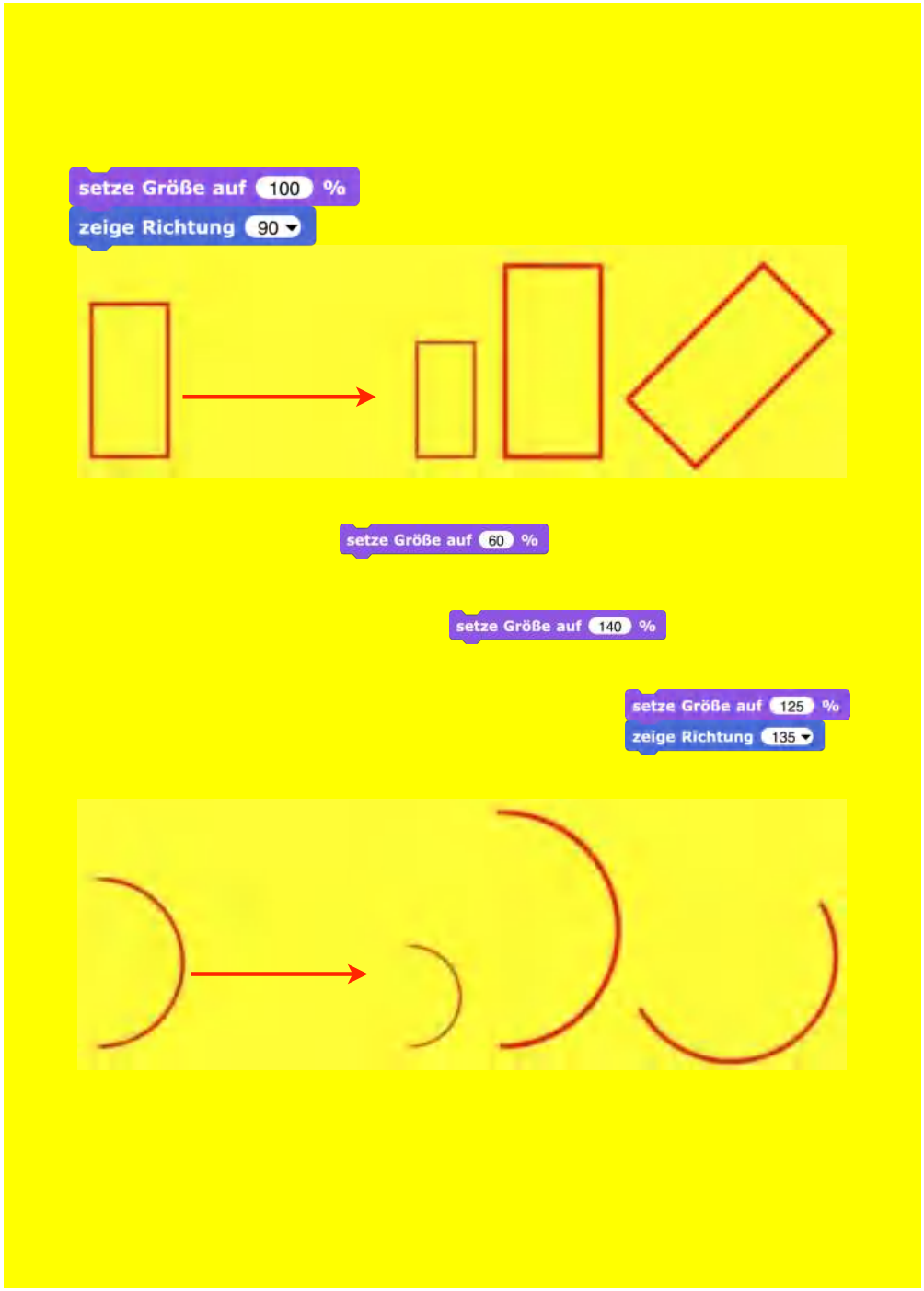


Im Programmbereich findet sich dann unter dem Kartenreiter **Kostüme** eine Kostümliste.

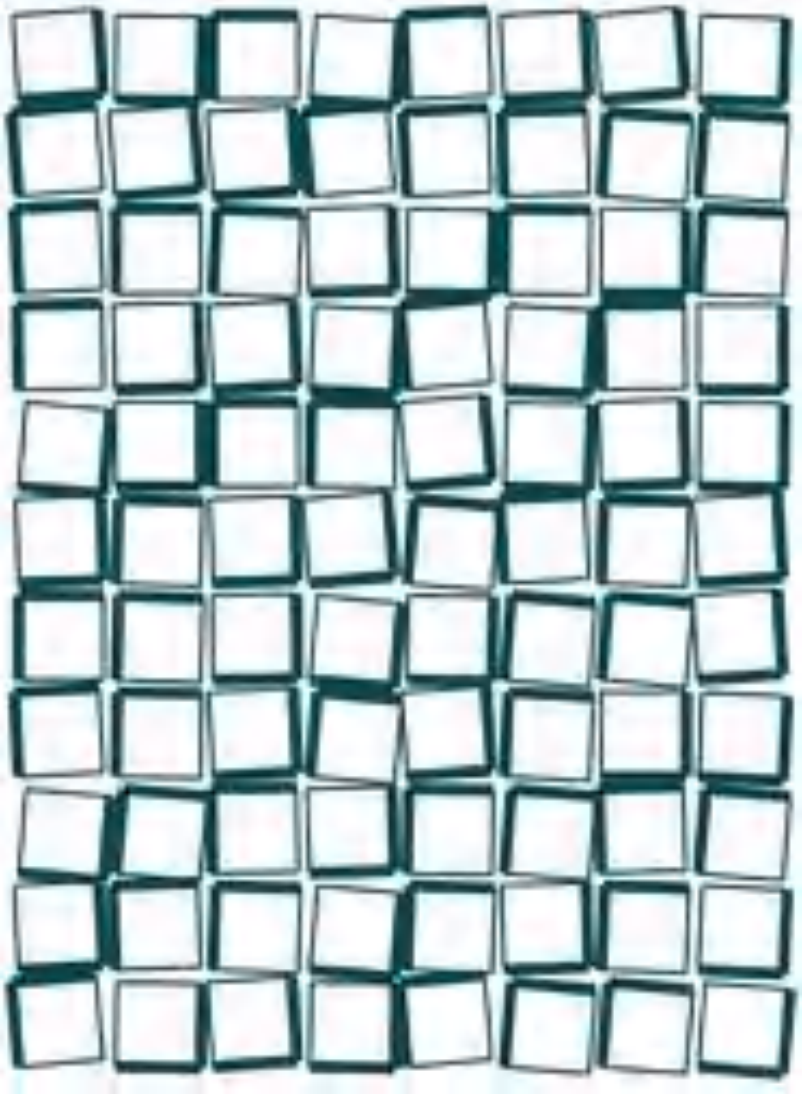
In der Kategorie **Aussehen** stehen Befehle für eine Anpassung der Kostüme zur Verfügung. Mit dem Befehl **ziehe Kostüm an** wird das aktuelle Kostüm der Schildkröte durch das ausgewählte Kostüm ersetzt. Mit **nächstes Kostüm** wird das aktuelle Kostüm durch das jeweils nächste Kostüm aus der Kostümliste ersetzt. Über **setze Größe auf x %** kann die Größe des Kostüms prozentual, mit **ändere Größe um x** um x Bildpunkte verändert werden.

Entscheidend ist nun, dass mit **stemple** das gewählte Kostüm an der jeweils aktuellen Position der Schildkröte „gestempelt“ werden kann. Es verbleibt dort, auch wenn die Schildkröte weiter bewegt wird.

Damit wird es möglich, einfacher und wesentlich schneller komplexe Grafiken zu vervielfachen als durch die entsprechenden Prozeduren in **wiederhole**-Schleifen. Die folgenden so entstandenen Bilder können das beispielhaft zeigen.

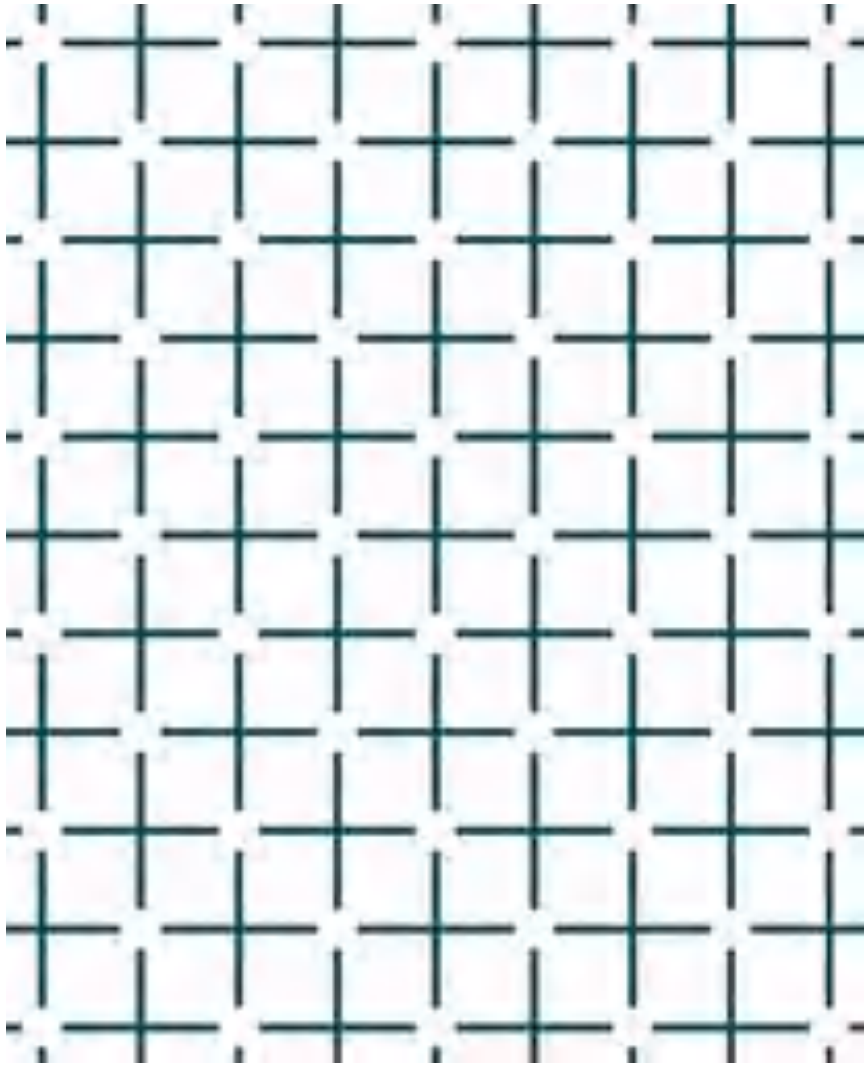


Quadrate



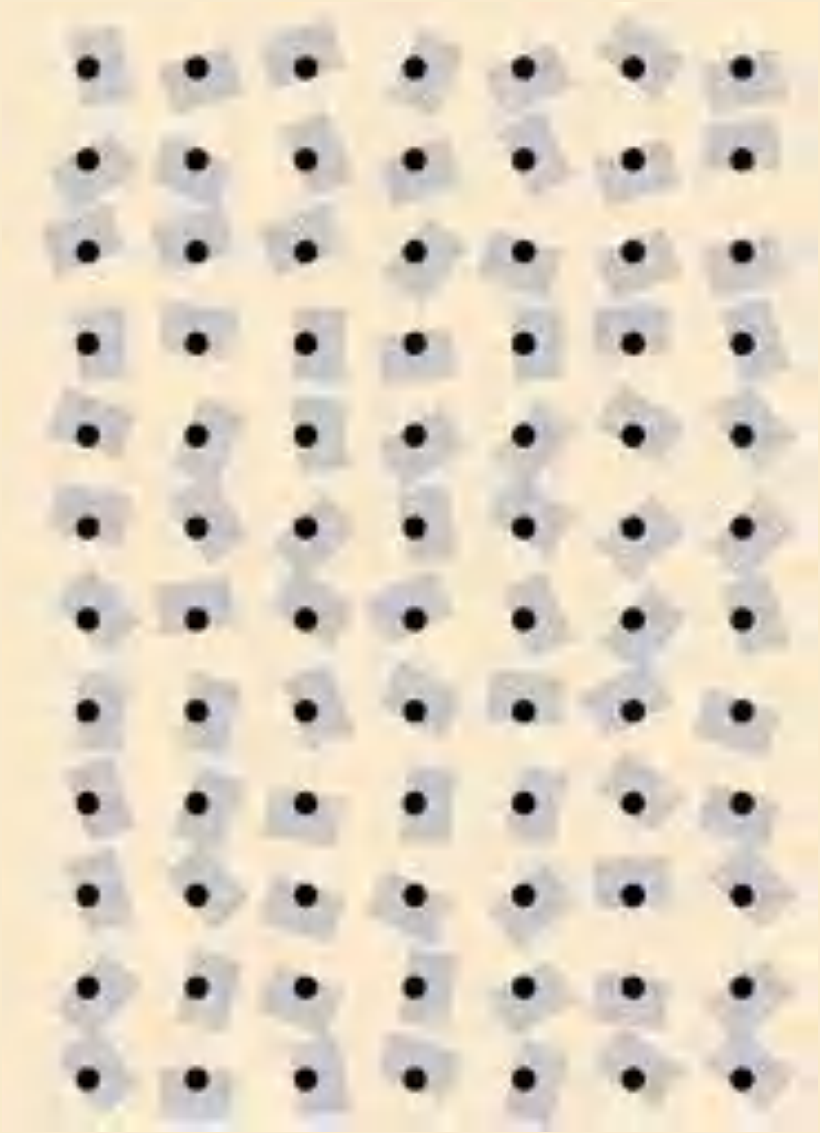
Hommage à Mohr: P-105

Linien



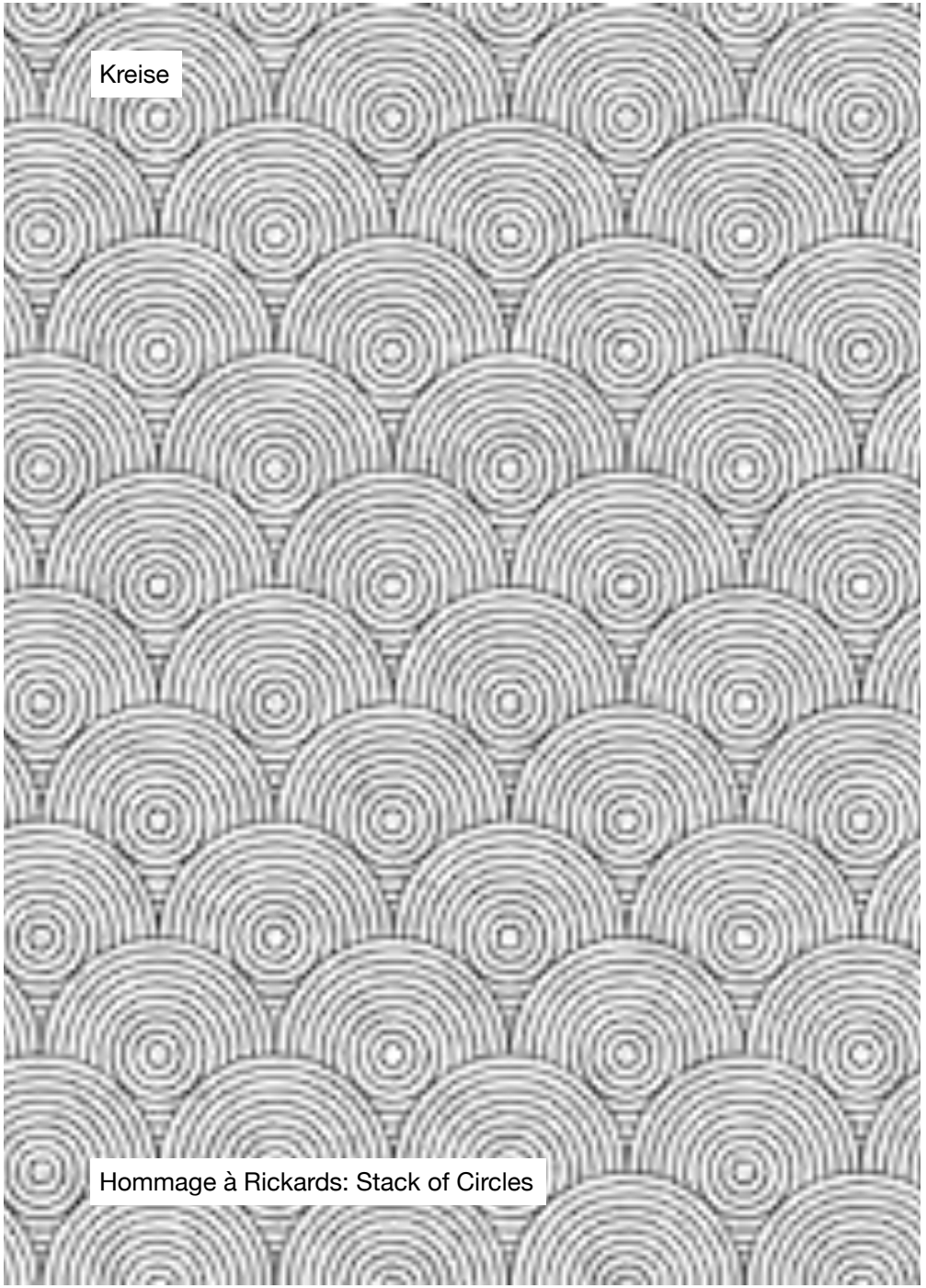
Opticals: Ehrenstein-Täuschung

Linien und Punkte



Hommage à Le Parc: Rotations

Hommage à Le Parc: Dots

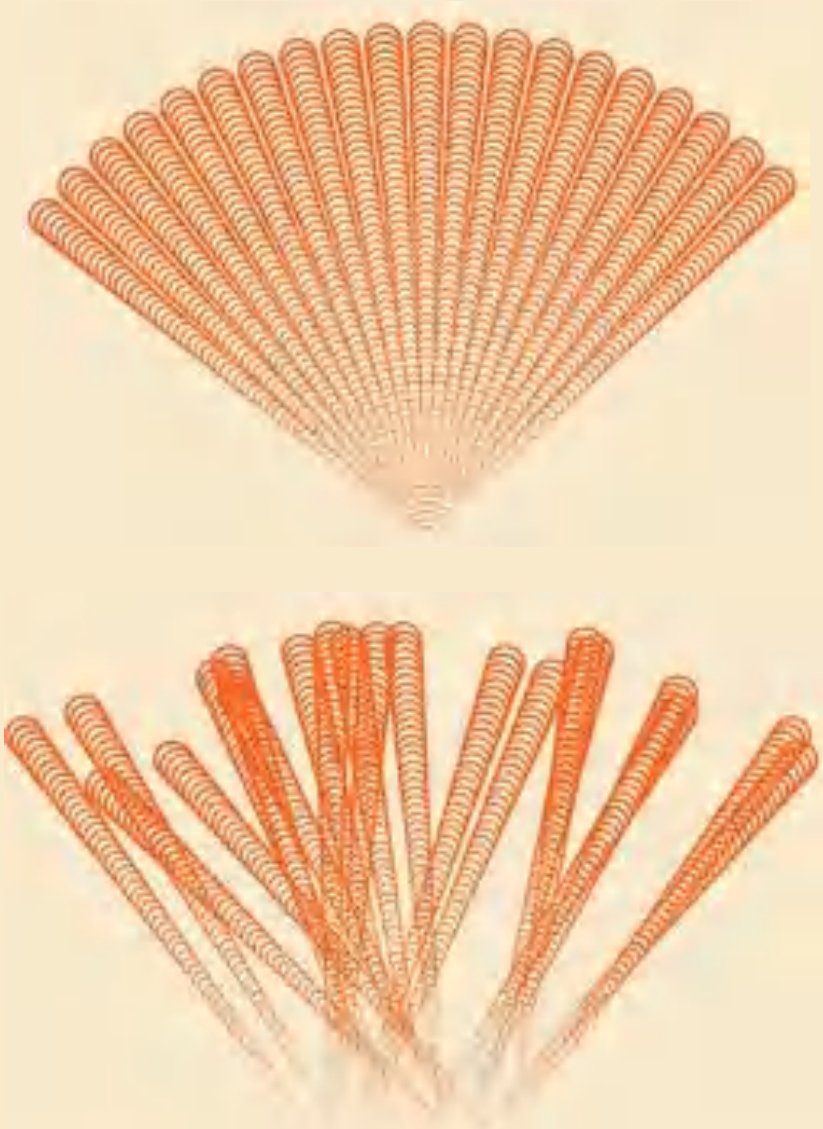


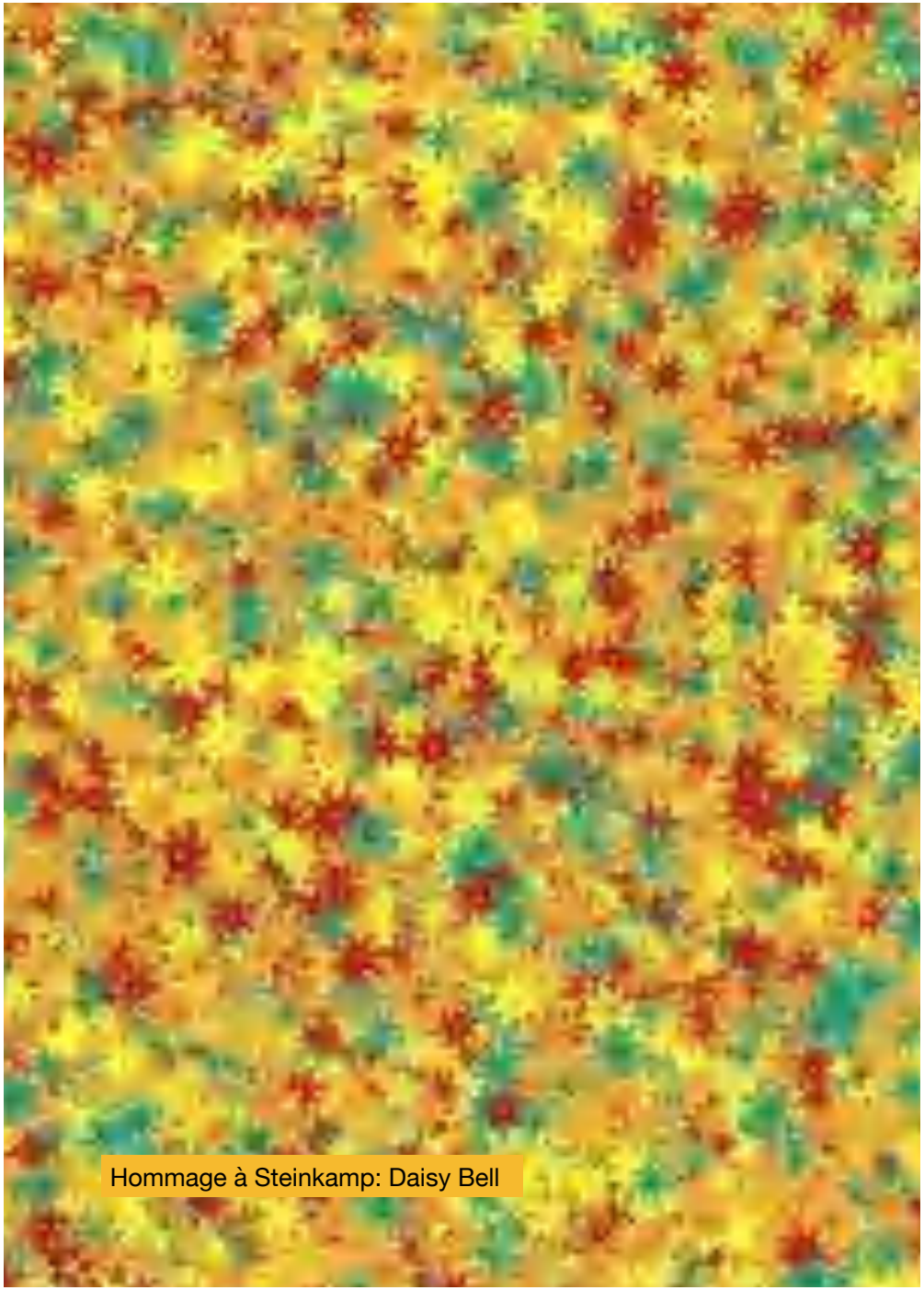
Kreise

Hommage à Rickards: Stack of Circles

Rechtecke

Kreisbögen





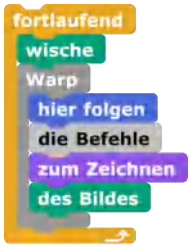
Hommage à Steinkamp: Daisy Bell

Ausblick: Animation und Interaktion

Den vollständigen Code der Beispiele finden Sie auf der Website Logo Klassiker. Dort führt ein Klick auf den **Programmnamen** jeweils direkt zu den Programmen in der Snap!-Programmierumgebung.

Manche der Programme unterscheiden sich dabei etwas von der im Text gezeigten Abfolge. Das liegt daran, dass ich die Programme - soweit sinnvoll - für die Animation vorbereitet habe. Weiterhin sind diese Programme auch interaktiv, d.h. sie können sehr einfach durch die Nutzer gesteuert werden. Die Vorgehensweise ist dabei immer die gleiche und leicht zu verstehen:

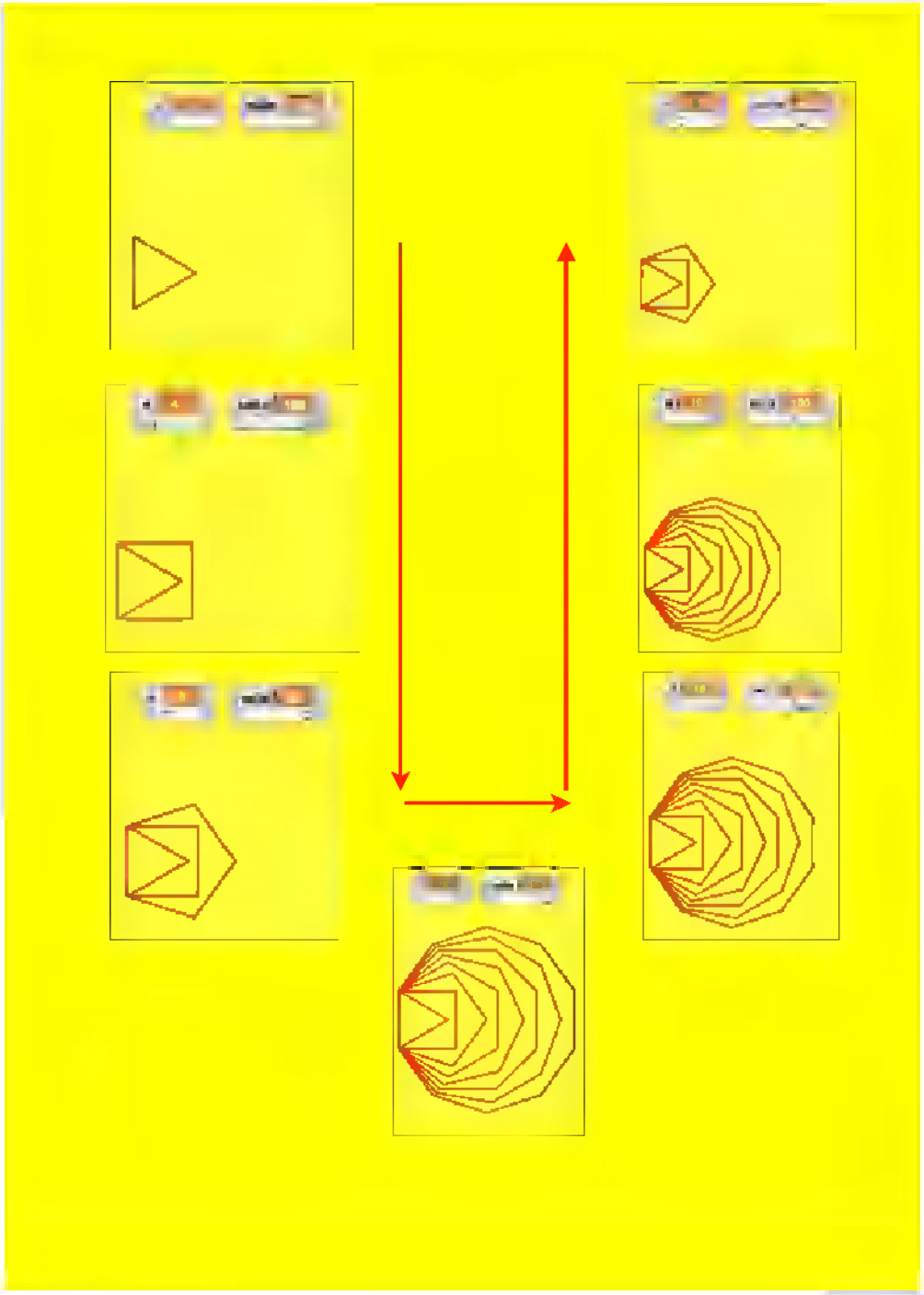
Das Animationsprinzip ist sehr einfach und entspricht der traditionellen Stop-Motion-Technik. Jedes Bild wird vom Computer gezeichnet. Das aktuelle Bild wird kontinuierlich durch ein neues Bild mit aktualisierten Eigenschaften (wie Farbe, Länge, Winkel usw.) ersetzt. Es ist also das Prinzip **Malen - Wischen - Malen ...** Wenn das schnell genug geschieht, erhalten wir eine flimmerfreie, animierte Darstellung.



Die Animation findet in einer Endlosschleife (**fortlaufend**) statt. Zuerst wird die Bühne gelöscht (**wische**), gefolgt von den **benötigten Befehlen zum Zeichnen des Bildes**. Der **Warp**-Befehl ist notwendig, weil er dafür sorgt, dass die grafische Ausgabe erst dann erfolgt, wenn das Bild vollständig gezeichnet ist. Das macht die Blldausgabe flüssiger!

Damit die Betrachter die Bilder interaktiv steuern können, sind für die Kennwerte **Schieberegler** vorgesehen. Werden damit Werte geändert, wird das Bild bei der nächsten Wiederholungsschleife mit diesen Werten aktualisiert. Um unerwünschte Effekte zu vermeiden, sollten sinnvolle **Minimal**- und **Maximalwerte** für die Kennwerte voreingestellt werden.

Im konkreten Beispiel auf der rechten Seite werden für die Polygone dynamisch die Zahl der Ecken und deren Seitenlänge fetsgelegt (hier nur statisch darstellbar).



Literatur

Abelson, H. (1983). Einführung in Logo. München: IWT-Verlag.

Eyferth, K., Fischer, K., Kling, U., Korte, W., Laubsch, J., Löthe, H., Schmidt, R., Schulte, H. & Werkhofer, K. (1974). Computer im Unterricht. Formen, Erfolge und Grenzen einer Lerntechnologie in der Schule. Stuttgart: Klett.

Feurzeig, W. (2010). Toward a Culture of Creativity: A Personal Perspective on Logo's Early Years and Ongoing Potential. International Journal of Computers for Mathematical Learning, Vol. 15, 3, pp. 257-265.

Harvey, B. (1997). Computer Science Logo Style. V. 1: Symbolic Computing. Cambridge: MIT Press. (Download des Buches: https://people.eecs.berkeley.edu/~bh/v1-toc2.html)

Hoppe H.-U. & Löthe, H. (1984). Problemlösen und Programmieren mit Logo. Ausgewählte Beispiele aus Mathematik und Informatik. Stuttgart: Teubner.

Nicolai, C. (2010). moiré index. Berlin: Die Gestalten Verlag.

Papert, S. (1980). Mindstorms: children, computers, and powerful ideas. New York: Basic Books. (Download des Buches: http://worrydream.com/refs/Papert%20-%20Mindstorms%201st%20ed.pdf)

Papert, S. (1982). Mindstorms. Kinder, Computer und Neues Lernen. Basel: Birkhäuser.

Schattschneider, D. (1990). Visions of Symmetry. New York: Freeman and Company,

Ziegenbalg, J. (1985). Programmieren lernen mit Logo. München: Hanser. (Download des Buches: http://www.ziegenbalg.ph-karlsruhe.de/materialien-homepage-jzbg/mybooks-scans/Programmieren-lernen-mit-Logo.pdf)

Bildernachweise:

Buchumschlag Mindstorms (S. 1): Foto J. Wedekind

Bodenturtle (S. 1): http://cyberneticzoo.com/cyberneticanimals/1969-the-logo-turtle-seymour-papert-marvin-minsky-et-al-american/

Alle anderen Bilder sind Screenshots aus den jeweiligen Programmen bzw. damit erzeugter Ergebnisgrafiken.

„Die Schildkrötengeometrie ist ein anderer Stil der Geometrie. Euklids Stil ist ein logischer, Descartes' Stil ist ein algebraischer. Schildkrötengeometrie ist ein algorithmischer Stil der Geometrie." (Papert, 1980)

Dieses Buch stellt typische mit der Schildkrötengrafik erzeugte Bilder in den Mittelpunkt. Sie finden sich vielfach in frühen Publikationen zu Logo (Handbüchern und einführenden Büchern) und sie entwickeln eine ganz eigene Ästhetik.

Anhand von Code-Beispielen in der visuellen Programmierumgebung Snap! kann die Entstehung dieser Bilder leicht nachvollzogen werden. Wer Freude an abstrakt-geometrischen Grafiken hat, wird vielfältige Anregungen finden, wie mit einfachen programmiertechnischen Mitteln entsprechende Bilder erzeugt werden können.