generative art

a practical guide using processing

SAMPLE CHAPTER

matt pearson

foreword by marius watz

MANNING

table of contents

Part 1 Creative Coding

1 Generative Art: In Theory and Practice

2 Processing: A Programming Language for Artists

Part 2 Randomness and Noise

3 The Wrong Way to Draw A Line

4 The Wrong Way to Draw a Circle

5 Adding Dimensions

Part 3 Complexity

6 Emergence

7 Autonomy

8 Fractals

Chapter One

1

Generative Art:
In Theory and Practice

Sample Chapter

MANNING

Chapter One

1

Generative Art: In Theory and Practice

On first appraisal, the question "What is generative art?" may seem simple. But, as is the nature of all things generative, even this definition has an emergent complexity.

The elucidation most often cited in recent years is attributed to Philip Galanter, artist and professor at Texas A&M University, from his 2003 paper "What Is Generative Art? Complexity Theory as a Context for Art Theory":[1] "Generative art refers to any art practice where the artist uses a system, such as a set of natural language rules, a computer program, a machine, or other procedural invention, which is set into motion with some degree of autonomy contributing to or resulting in a completed work of art."

Although this is accurate and descriptive—and a long sentence with all the right words—a single phrase like this isn't enough. I don't think it quite captures the *essence* of generative art (GenArt), which is much more nebulous. In my mind, GenArt is just another byproduct of the eternal titanic battle between the forces of chaos and order trying to work out their natural harmony, as expressed in a ballet of light and pixels. But flowery crap like that isn't going to get us anywhere either

1. You can read Prof. Galanter's paper in full at www.philipgalanter.com/downloads/ga2003_paper.pdf.

We have to be careful treading around this topic, because we want at all costs to avoid trying to define "What is art?" which is an argument best left alone. The concept of art can be so fragile and fuzzy that if we were to prod it too much, it would evaporate. So, instead of trying to carve up a subject that doesn't want to be dissected in search of a pithy description, in the section that follows I'll take a more delicate and obtuse approach. We'll begin by examining what generative art *isn't*.

1.1 Not your father's art form

With more traditional art forms—sculpture, painting, or film, for example—an artist uses tools to fashion materials into a finished work. This is clearly doing it the hard way. With generative art, the autonomous system does all the heavy lifting; the artist only provides the instructions to the system and the initial conditions.

The artist's role in the production process may be closer to that of a curator than a creator. You create a system, model it, nurture it, and refine it, but ultimately your ownership of the work produced may be no more than a parent's pride in the work of their offspring.

This is hideously unfair, of course. We shouldn't underestimate the human role in the collaboration. In addition to the programming, the human contributes one other important skill: *aesthetic judgment*. It's feasible for computers to develop a sense of aesthetics (plenty of work toward this aim has been done within the field of evolutionary systems[2]), but it will never be the best division of labor in a human-machine creative partnership. If we need to calculate pi to a million digits, it would be a misappropriation of resources to set a human brain to this task. Similarly, it's probably best not to leave it to machines to decide what's pretty and what isn't.

Although GenArt is almost always abstract in nature, it can't be defined by the style of the work. The common factor of generative artworks is the methodology of its production, not the style of the end result.

Figure 1.1, for example, is an abstract work, and it's a monochrome work, but we can only say that it's a generative work if it happens to fit my claims as to how it was created. This particular arrangement of pixels may have been achieved in Adobe Illustrator (not by my cloddish paws,

2. Richard Dawkins' book *The Blind Watchmaker* (1986) is a good, accessible introduction to the topic of genetic algorithms and their application.

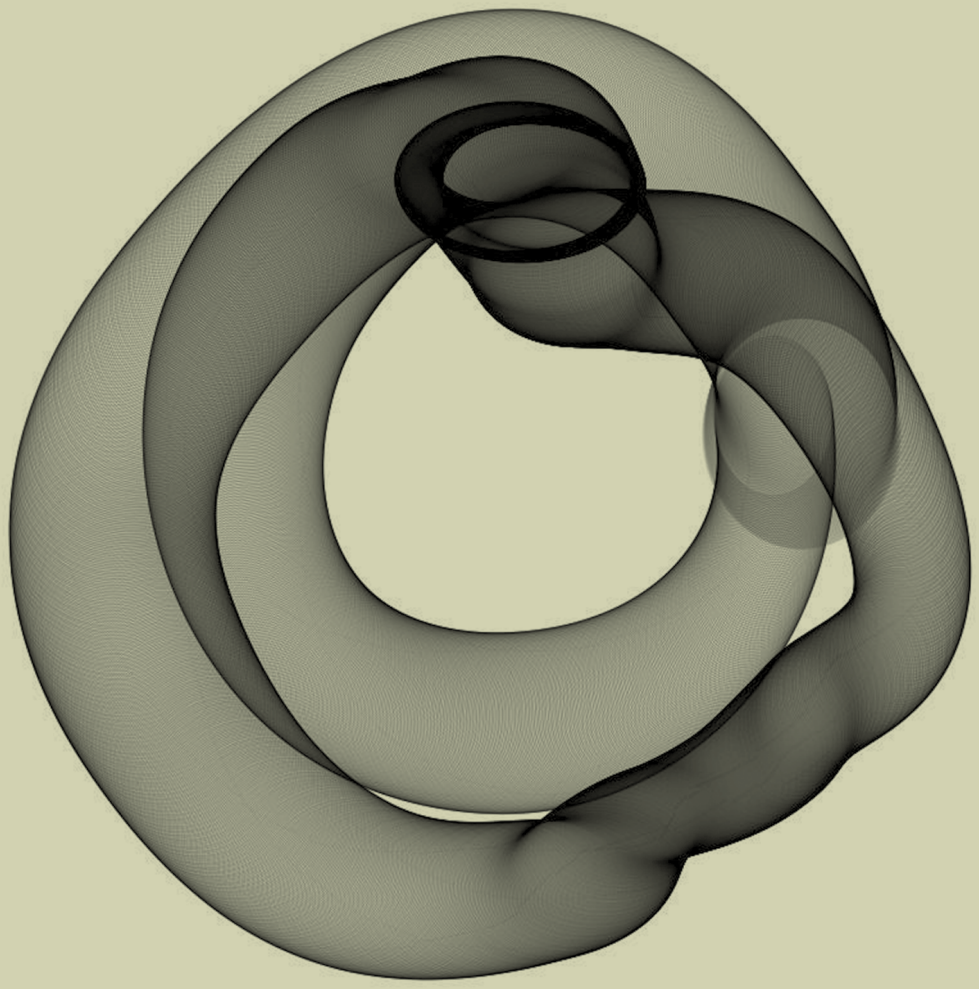

Figure 1.1
Tube Clock (2009)

I would hasten to add); by photography; or by pencil, paper, and scanner; and in all cases, it would still be a monochrome abstract. It may also still be generative, depending on the way each of these tools is used, but this isn't a given.

The tools used aren't the defining factor; it's the *way* they're used that provides the commonality. In this book, the programming language, specifically the Processing language, is the chosen tool. But that doesn't mean everything you create with that tool is generative. Programming languages are just ways of making computers do as they're told; there is nothing inherently generative about following orders.

To be able to call a methodology *generative*, our first hard-and-fast rule needs to be that *autonomy must be involved.* The artist creates ground rules and formulae, usually including random or semi-random elements, and then kicks off an autonomous process to create the artwork. The system can't be entirely under the control of the artist, or the only generative element is the artist herself. The second hard-and-fast rule therefore is *there must be a degree of unpredictability.* It must be possible for the artist to be as surprised by the outcome as anyone else.

Creating a generative artwork is always a collaboration, even if the artist works alone. Part-authorship of any generative work must belong in part to the mechanisms the artist uses: the system that generates it. Fortunately, anonymous autonomous systems aren't usually bothered if their unscrupulous artistic partners decide to steal all the credit.

To retain a necessary focus, this book discusses only one tool—the programming language—as a way of producing only one range of output: visuals. But generative methods may also be used to produce music, architecture, poetry, dance, storytelling or interactive experiences, and the autonomous systems behind their creation may also be mechanical, games of chance, natural phenomena, or subconscious human behavior. We'll touch on some of these alternative approaches later in the book.

1.2 The history of a new idea

Generative art has a history measured in decades, not long compared to other arts, which is probably why it's still on the periphery of the art world. Art colleges across the globe are churning out tens of thousands of painters, potters, fashion designers, and graphic designers every year, but the number of practicing generative artists in the world at present could probably fit comfortably onto a single Caribbean cruise liner (which would be a lovely idea if anyone fancies arranging it). This demographic is changing fast, though. As popular computing technology accelerates, more creative people are getting theirs hands on the tools and discovering this novel art form.

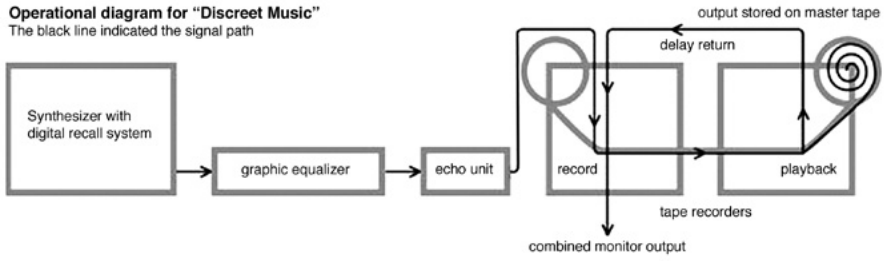

Figure 1.2

Brian Eno's *Discreet Music* album included a diagram on the sleeve explaining the synthesizer and tape-loop feedback system by which the music had been produced.

The term *generative art* has only been in general use since the 1960s, but the concept has been with us much longer. Generative forms of music, for example, have been around since Mozart. His *Musikalisches Würfelspiel* (Musical Dice Game) was an early example of a generative artistic system.

The idea was to create a minuet by cutting and pasting together prewritten sections, making selections according to the roll of a die. Even with a single six-sided die, the number of possible combinations rises quickly: by 5 rolls, there are 7,776 possible combinations; and with 6 rolls, 46,656. These types of artistic parlor games became popular in the eighteenth century.

In the last century, composers such as John Cage, Karlheinz Stockhausen, and Brian Eno expanded on the idea of generative music.[3] John Cage's *4' 33",* his controversial note-less piece defined only by its length, takes environmental ambient sounds as its only content, meaning no two performances of the work are ever the same.

Later, Stockhausen and Eno (and others) experimented with procedural methods of composition, where music is defined by a set of rules or conditions. Eno's *Discreet Music* LP (1975) is a fine example of this. The first side is a 30-minute piece created by a tape-loop feedback system. A synthesized melody was recorded onto a tape machine, the output of which was fed into a second tape machine. The output of the second machine was then fed back into the first machine and the overlapping signals recorded. We know of this process because Eno included a diagram of his setup on the back cover of the LP (see figure 1.2).

3. Even though Processing can be used to create generative music, this book focuses solely on the creation of visuals. Generative music is a huge and fascinating topic that warrants a book of its own. David Toop's *Haunted Weather* might be a good place to start.

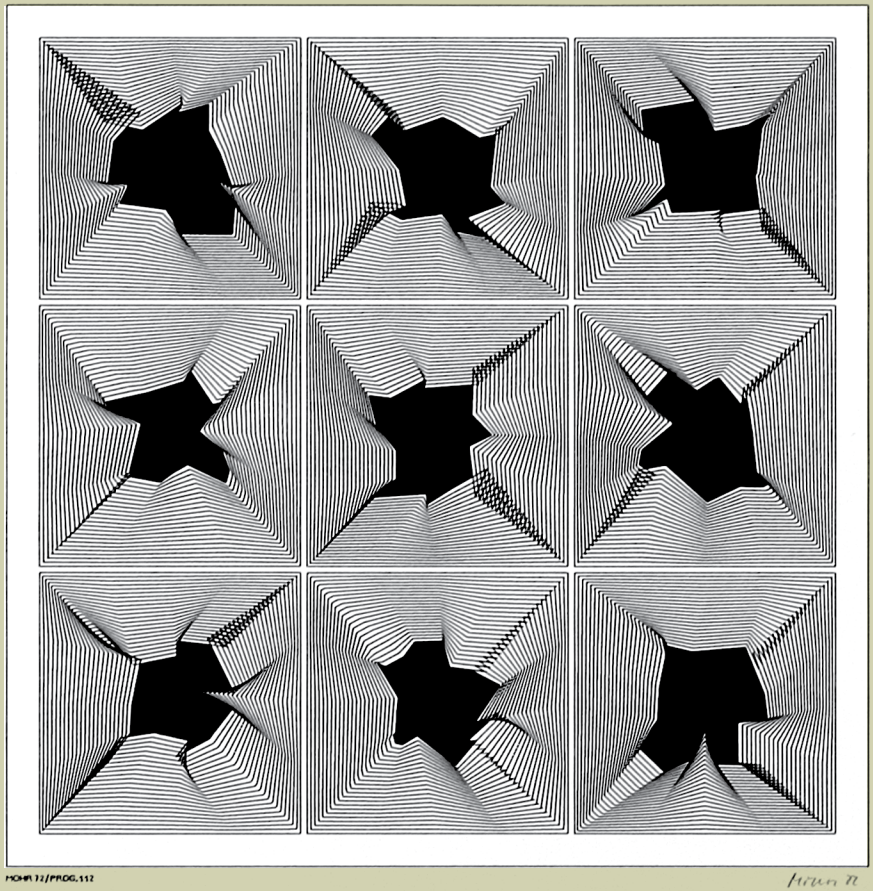

Figure 1.3

Lady Quark by Manfred Mohr (1972) , one of the earliest *algorists* (generative artists who work with computers). Mohr wrote his own software in order to explore his art.

Visual forms of generative art started emerging in the 1960s, first with computers outputting to plotters, then with visual display units (VDUs), and later in more sophisticated forms of print and video. Early pioneers from the plotter years were Frieder Nake, George Nees, Vera Molnar, Paul Brown, and Manfred Mohr (see figure 1.3), who published a collection of computer-generated artworks called *Artificiata I* in 1969.

But although the development of GenArt has been closely tied to the evolution of the computer, computers are just a useful convenience. The *real* tools of GenArt, the underlying constants to the various tools you can use, are the *algorithms*. Algorithms are a part of the natural world; they have a universality that transcends medium. So while the systems capable of creating GenArt change over time, evolving as technology evolves, the algorithms remain the same. Generative art may not be quite as old as art itself, but it may be said to be at least as old as mathematics.

This book is all about the algorithms. It's about the philosophy, aesthetics, and experimentation too, but mostly it's about the algorithms and the tools you use to explore them. You can, theoretically, perform the mathematics behind procedural GenArt with a stick of chalk and a large flat rock, but that would make life unnecessarily difficult—especially when we live among the ever-advancing sophistications of the twenty-first century and its remarkable digital toolset, which is what we'll look at next.

1.3 The digital toolset

If you're a practitioner within the digital medium (and it's difficult not to be these days), you likely already have a set of tools you favor for your creative endeavors. You've come to grips with a particular word processor or email client, so when you feel the need to craft words, you have the means to do so. You may also have invested time in learning a graphics package for when you're inclined to say it with pixels. On a more fundamental level, you've probably developed relationships, perhaps even dependencies, with your devices—MP3 players, phones, laptops—and have tamed them into serving your needs. Software and hardware alike, all of these are the tools in your digital toolbox. But how long-term are these man-machine marriages?

1.3.1 Perpetual impermanence

In periods of rapid technological innovation (of which our current time is one, but so has been pretty much every other era in human history) we become highly focused on the latest tools. We invest a lot of time in mastering their quirks, and we upgrade them frequently.

Figure 1.4

The ZX Spectrum: the machine I learned to program on. It was the cutting edge of British home computing circa 1982: 48 KB RAM, no hard disk, and no mouse. The lack of a visual operating system meant you had to learn basic programming skills just to be able to load a game.

Apple's range of laptops is updated annually, with the top of the range reaching higher and higher levels of power and convenience. An application like Adobe Photoshop has, on average, a major new release every 18 months, with a dollar price for each release in the hundreds. It would be unimaginable to buy a chisel, or a paintbrush, or a set square, knowing that it would be obsolete in two years, yet this almost-immediate obsolescence is unquestioningly accepted on the cutting edge of both hardware and software.

An unwritten complicity exists between users and manufacturers of technology. The manufacturer will have a roaring trade if every year it can deliver the same product to the same customer who bought it last year, but this is made possible only by the demand of the market: the users who eagerly await these upgrades and see value in every minor iteration. They're driven by the desire for the latest technology has to offer, and they're prepared to make the necessary sacrifices to have it within their grasp. This creates a cycle of perpetual impermanence, which marches the technology forward.

You should bear this in mind as you create digital artwork. If you work with pixels, there is already a fragility to the work you do: your art disappears the second the monitor is turned off. For this reason alone, many artists gravitate toward more tangible art forms such as print and sculpture (the decreasing prices of 3D printers promise some exciting possibilities for tangible generative art in the coming decade). Furthermore, if you become dependent on a machine to create your art, you know the machine and the software it runs are already halfway to obsolescence before you've taken them out the box. This isn't a bad thing, though, because the next generation will be capable of even more remarkable mathematical feats, enabling ever-broader artistic possibilities.

From our limited temporal perspective, the evolution of digital tools may seem to be in a uniformly positive direction, with every release bringing greater sophistication, speed, simplicity and/or intuitiveness. Moore's Law; the 1960s observation that processing power is doubling, and its price halving, every few years, seems to be holding true and showing no sign of dropping away—which, for now, means the digital world is only ever going to get Faster! Better! Cheaper! But as we're swept forward on this breakneck digital tide, we may fool ourselves into believing that our functional friends are at an unprecedented level of sophistication, that they're the best our civilization can offer. But by what yardstick do we measure such sophistication?

1.3.2 The latest in primitive technology

We're in the very early days of digital technology. The computer is still in its first century—it's but an infant, and the art of human-computer interaction is still a work in progress. Applications such as Photoshop, Flash, GarageBand, After Effects, Illustrator, Final Cut Pro, Maya, syntaxes such as C++, Java, HTML, CSS, and so on, all do incredibly cool things and have changed the way our media looks and behaves. But they aren't tools with the simplicity or intuitiveness of the paintbrush or the scalpel. They're clearly much removed from pencils, pens, chisels, guitars, drums, trowels, and scissors, from natural and instinctual ways of expressing ourselves that have evolved over the millennia, rather than the last few decades. Compared this way, our digital tools appear extremely primitive in their usability, and at times quite an impediment to our creative flow. The purpose of a tool is not only to extend our capabilities; it should also enhance the flow of our creativity. The newest software gives us the power to fashion the world in ways that would have been unimaginable only 50 years ago, but there is still a long way to go in terms of a natural artistry to their use. Almost all of the more powerful tools have a steep learning curve and require dedication and constant use to master. A child can't quickly pick up Adobe Illustrator and draw a line. Even adults with a certain degree of techno-savvy can't come fresh to a program like Maya and find a way to express themselves within a few minutes. On the other hand, using a pencil or

a paintbrush, a child could instantly make some kind of statement involving line and color without having to learn any keyboard shortcuts. Their efforts would likely be rather amateur, it's true, but the important point is that they haven't been at all hindered by their tool, only empowered by it.

An art form is defined by its tools. The tools give an art form its grammar. If the nature of a tool means it's easy to perform one action or express one idea, and harder to perform another, the easier option will become more common. This is why so few landscape painters work in 3D, and why choreographers aren't interested in bowls of fruit. In this way, our tools have a greater influence on the work they're put into producing than you may realize. They can limit our creative choices at the same time they extend them.

The trick is to appreciate these parameters and to come up with new ways of using the tools, to avoid falling into tired and over-familiar practices. This applies to a programming language as much as it does a paintbrush. If you're an experienced programmer with a more formal grounding in your trade, that is one of the things I hope you'll get from this book: a fresh way of approaching the art of coding. And if you've never coded before, I hope you'll find novelty in these new toys. For what use is a paintbrush without an artist with the will to hold it?

1.4 Summary

We've explored what generative art *isn't*, and we've grudgingly accepted the limits of our primitive and impermanent toolset. Surely there must be something positive to take away from this chapter.

For starters, we've built a list a few things that, it's probably safe to say, generative art definitely *is*. It's:

- An algorithmic way of creating an aesthetic
- A collaboration between an artist and an autonomous system
- An exercise in extracting unpredictable results from perfectly deterministic processes
- A quest for that sweet spot between order and chaos
- A fresh, fun approach to coding
- A growing medium with huge potential

It may be early days for the digital toolset, but these technological marvels award us some novel powers of expression. At the same time, they have a huge bearing on the type of work we produce with them. The tool we'll focus on for the GenArt experiments in this book is a wonderfully simple language called Processing, which we'll get started with in the next chapter.

DIGITAL DESIGN/PROGRAMMING

generative art

pearson

Artists have always explored new media, and computer-based artists are no exception. Generative art, a technique where the artist creates print or onscreen images by using computer algorithms, finds the artistic intersection of programming, computer graphics, and individual expression.

Generative Art presents both the techniques and the beauty of algorithmic art. In it, you'll find dozens of high-quality examples of generative art, along with the specific steps the author followed to create each unique piece using the Processing programming language. The book includes concise tutorials for each of the technical components required to create the book's images, and it offers countless suggestions for how you can combine and reuse the various techniques to create your own works.

What's inside

- The principles of algorithmic art
- A Processing language tutorial
- Using organic, pseudo-random, emergent, and fractal processes

Matt Pearson is an artist, coder, and award-winning blogger.

FREE eBook
SEE INSERT

For access to the book's forum and a free ebook for owners of this book, go to manning.com/GenerativeArt

"Perfectly placed at the intersection of code and creative thinking."

From the Foreword by
Marius Watz
Founder of Generator.x

"Succeeds in teaching without lecturing ... turns dull programming concepts into fun explorations."

Frederik Vanhoutte
Generative Artist, wblut.com

"If you're interested in generative art, this is the place to start."

Robert O'Rourke
Founder of HasCanvas

US $39.99 / Can $45.99 (including eBook)

ISBN-13: 978-1-935182-62-7
ISBN-10: 1-935182-62-5
53999
9 781935 182627