generative art

a practical guide using processing

SAMPLE CHAPTER

matt pearson

foreword by marius watz

MANNING

table of contents

Introduction

The Organic vs. The Mechanical

Sample Chapter

MANNING

introduction: the organic vs. the mechanical

> From the standpoint of Taoist philosophy natural forms are not made but grown, and there is a radical difference between the organic and the mechanical.
>
> Things which are made, such as houses, furniture, and machines, are an assemblage of parts put together, or shaped, like sculpture, from the outside inwards.
>
> But things which grow shape themselves from within outwards—they are not assemblages of originally distinct parts; they partition themselves, elaborating their own structure from the whole to the parts, from the simple to the complex.

Alan Watts, 1958

Alan Watts (1915–73), English philosopher and Zen monk, was a Buddhist in a very 1960s sense. He was a master of theology, a priest, and the author of more than 20 books on Zen philosophy. He also experimented with psychedelic drugs, both on a personal level and in laboratory trials. He had plenty to say on the subject of creativity and technology but never, as far as I know, said anything specifically on the subject of generative art.

In the above quote, he's talking about the incongruity between the natural world and the manmade, separating creation into the organic and the mechanical. This concept of organic growth, whereby forms are constructed "from within outwards" describes this book's topic rather well; but in such a clear bilateralism, how can we say that a work of computer programming belongs to the world of the organic rather than the mechanical?

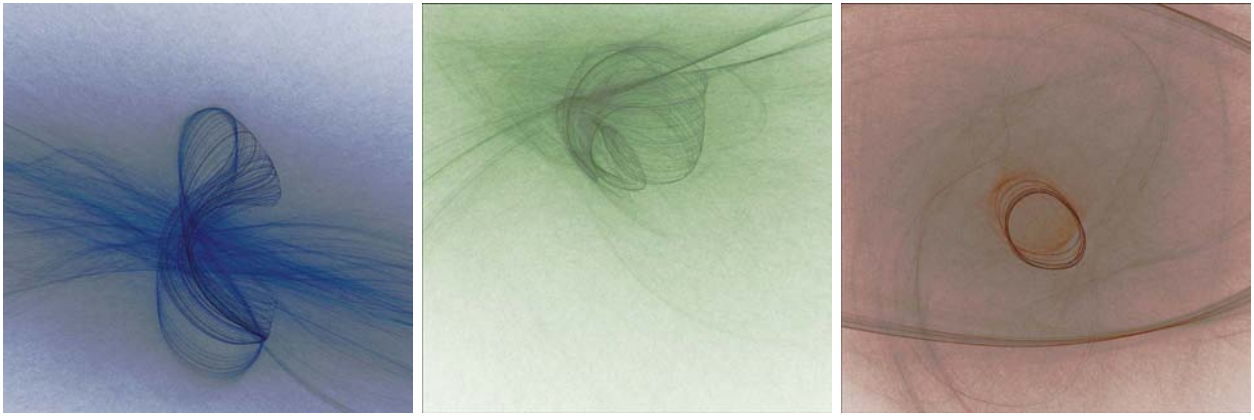
Generative art is neither programming nor art, in their conventional sense. It's both and neither of these things. Programming is an interface between man and machine; it's a clean, logical discipline, with clearly defined aims. Art is an emotional subject, highly subjective and defying definition. Generative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

Generative art isn't something we build, with plans, materials, and tools. It's grown, much like a flower or a tree is grown; but its seeds are logic and electronics rather than soil and water. It's an emergent property of the simplest of processes: logical decisions and mathematics. Generative art is about creating the organic using the mechanical.

Watts' opposing categories of worldly things, the mechanical and the organic, are easy to separate. A building has straight edges and sharp corners: it's functional and accurate; it's in the realm of the mechanical. A tree is irregular and temporally inconstant, its leaves shake in the wind and shed in the autumn; it's in the realm of the organic. Mechanical things are constructed; they're fashioned, as Watts says, from the outside in. They're built, drawn, assembled, sculpted, manufactured. On the other hand, organic things are grown: they're self-structuring, holistic. Their forms come about without intent; they don't conform to designs or blueprints.

Like the landscape gardener, the lot of the generative artist is to take naturally evolving phenomena and to fashion them into something aesthetically pleasing. It's finding that point of balance between the beautiful unruliness of the natural world and the desired order of our ape brains. A garden that is unkempt and overgrown is unpleasing to us because it's too far into the realm of the chaotic, whereas concreting the area instead is the tidiest, most ordered of solutions, which also removes all beauty.

The sweet spot is between the two, where the grass is neat and evenly cut but still no two blades are alike or move in perfect synchronicity—where the colors of the flowers are evenly balanced, but not in a way that is exact and precise. The sweet spot is where the "art" lives.



Figures i.1, i.2, and i.3

Opiamas Trangelo (2010) by Matt Pearson. Multiple artworks produced by a single algorithm.

From here on, if the artist is unspecified, assume that the artworks are by the author.

You may be used to skimming figure captions for clues as to how the figure relates to the chapter. But sometimes a figure is just there because it's prettier to look at than unbroken text. Sorry.

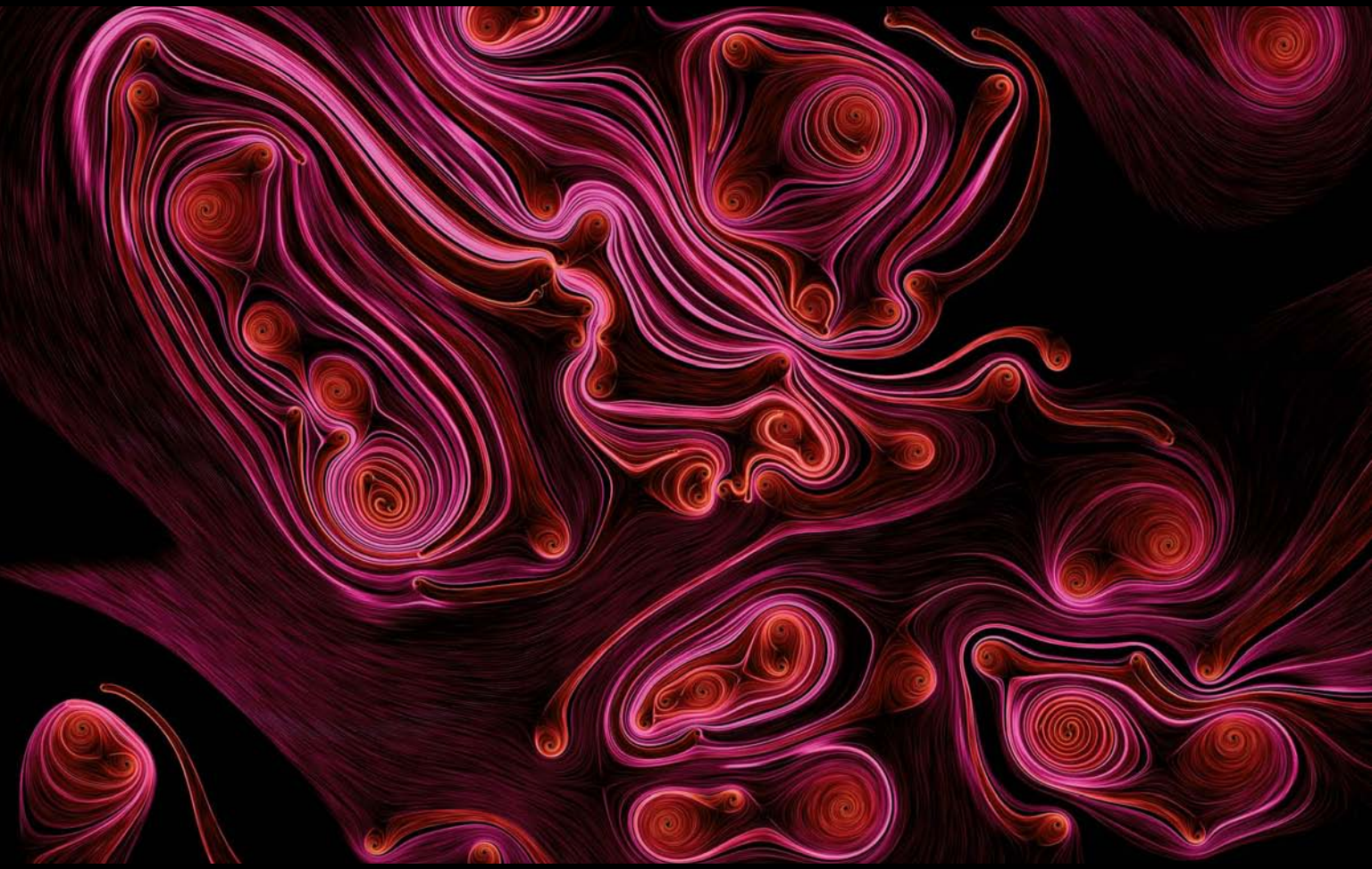


Figure i.4

Addition/Subtraction, Variant (2010) by Robert Hodgin, an artist whose work continues to be a huge inspiration. See also figures i.5 and i.6.

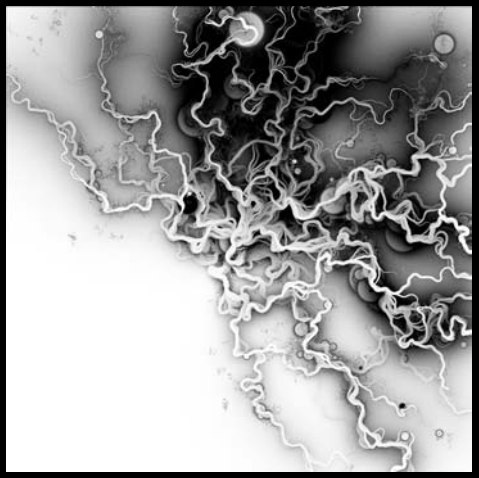


Figure i.5

Magnetic Ink by Robert Hodgin (2007). This work was created using a flocking algorithm, each agent leaving a mist of ink. See www.flight404.com/blog/?p=86 for further explanation. You'll learn more about flocking in chapter 6.

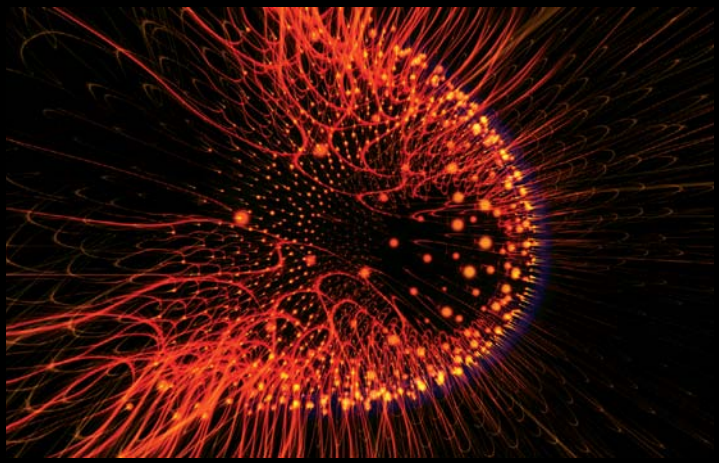


Figure i.6

Jelly (Magnetosphere) by Robert Hodgin (2007) . Most of Robert's early work, including this image, was created using Processing, a tool you'll be learning a lot about.

Generative art is easy

If you were to decide to become an artist, you would need to take only one step: write your name, write "(artist)" after it, and then print it on a business card. Congratulations: you're an artist. From this point on, every scribble, gesture, utterance, or movement you make can be defined as art, if you so choose. Every stool you leave in the cistern of life can find meaning to someone, somewhere, even if it's only yourself. You may not be a particularly *good* artist, but an artist is what you are.

Becoming a "generative artist" is a little more specific, but not that much more difficult. You may infer that some kind of skill set is required to qualify for this title. Perhaps you see learning a programming language as a significant barrier to entry. This isn't the case; the language isn't a barrier, it's a shortcut. Compared with other disciplines of the arts, where a minimum skill level is necessary for your work be taken seriously, with generative art most of the skill doesn't have to be learned; it's already encapsulated within the tools.

It takes many *years* to learn to paint, to draw, or to sculpt, but the programming aptitude required to get professional results in generative art can be learned in a matter of days. And if you don't believe me, I hope this book will prove it. The images in figure i.7 were generated in only 24 lines of code, which you can view in the listing on the next page. Every time it runs, it produces a different still image.

All this code does is iterate through a grid using two loops; then, a function call on line 14 draws a circle at each grid point and displaces it in 3D space using a mathematical variance. Don't worry if you don't understand the code, or indeed the previous sentence, at this point: that's what this book is for. We'll unpack all these concepts in the first few chapters.

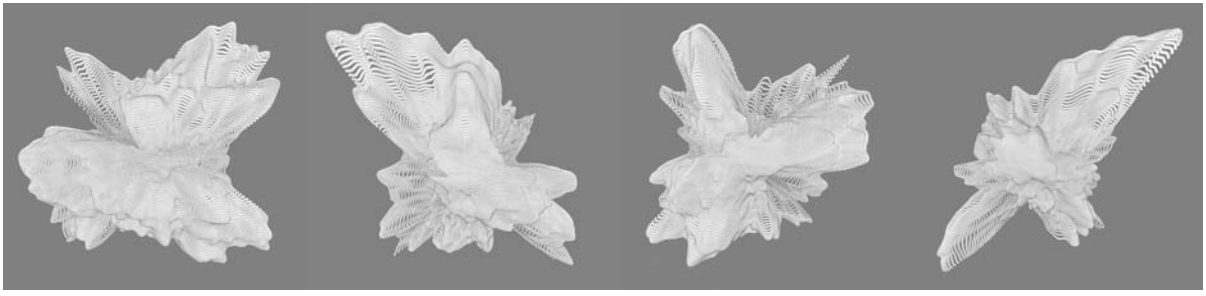


Figure i.7

Four generative works created by the 24 lines of code in listing i.1

Listing i.1 **A generative system in 24 lines of code**

```
void setup() {
  size(2000, 2000, P3D);
  background(150);
  stroke(0, 50);
  fill(255, 200);
  float xstart = random(10);
  float ynoise = random(10);
  translate(width/2, height/2, 0);
  for (float y = -(height/8); y <= (height/8); y+=3) {
    ynoise += 0.02;
    float xnoise = xstart;
    for (float x = -(width/8); x <= (width/8); x+=3) {
      xnoise += 0.02;
      drawPoint(x, y, noise(xnoise, ynoise));
    }
  }
}

void drawPoint(float x, float y, float noiseFactor) {
  pushMatrix();
  translate(x * noiseFactor * 4, y * noiseFactor * 4, -y);
  float edgeSize = noiseFactor * 26;
  ellipse(0, 0, edgeSize, edgeSize);
  popMatrix();
}
```

Even if you've never coded before, by the end of chapter 2 you'll have a powerful programming language under your belt. Then, from chapter 3 onward, you'll learn a number of techniques that will enable you to quickly achieve generative art proficiency. Taking this proficiency on toward mastery is then up to you.

If you come blessed with programming skills, you're already most of the way there. In fact, the main problem for experienced programmers may be unlearning the strict practices you're used to. Chaos is something we don't usually expect or welcome from computing devices, but for the next few hundred pages it will be our friend and collaborator. Order and chaos may be mutually exclusive, but that doesn't mean they can't work together.

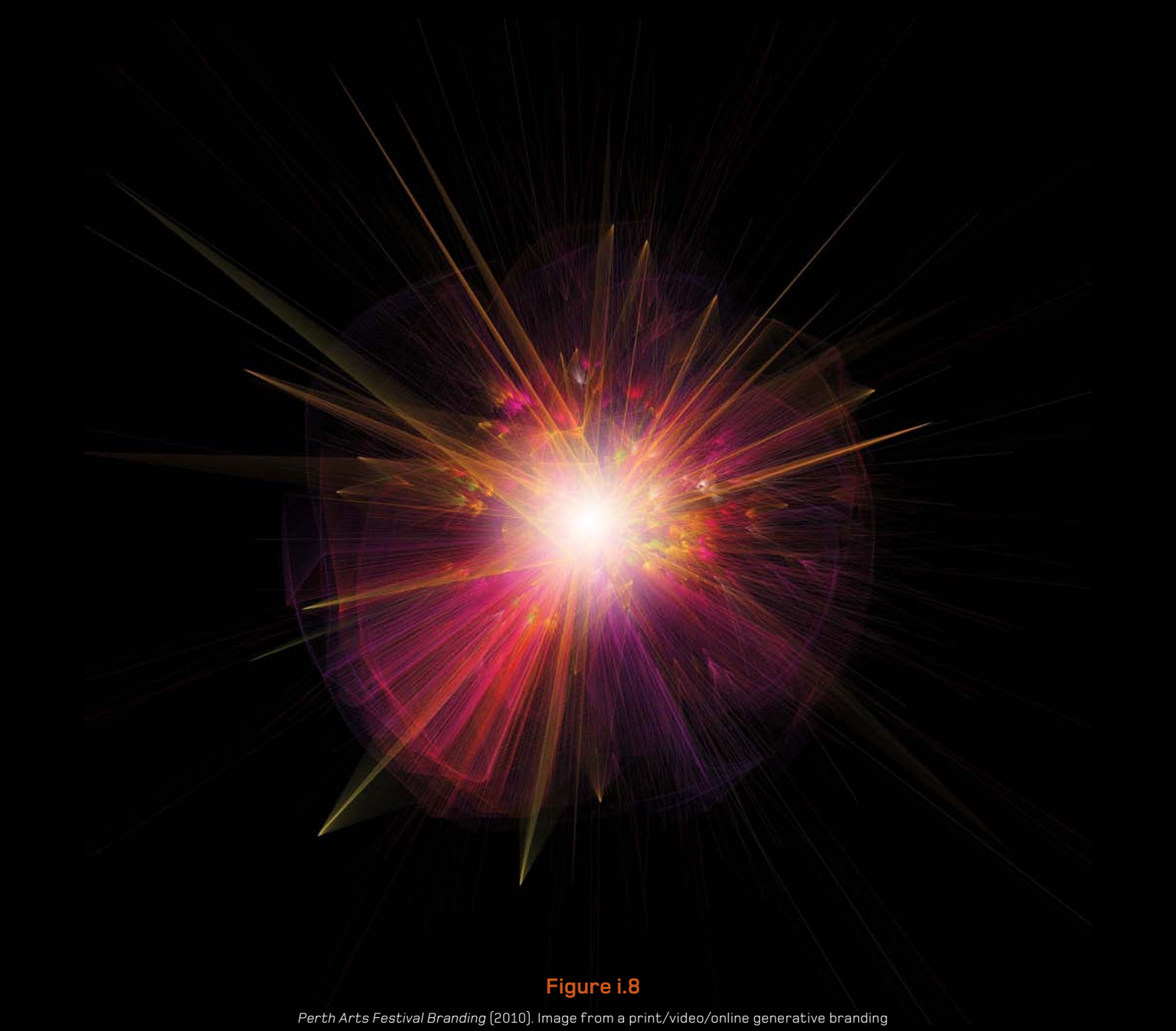


Figure i.8

Perth Arts Festival Branding (2010). Image from a print/video/online generative branding project I created in collaboration with Brighton's FutureDeluxe (http://futuredeluxe.co.uk). See http://vimeo.com/20096257 for the full story.

Order and chaos

The image in figure i.9 shows *Newton*, painted by William Blake in 1795. To the left of the picture are brightly colored flora and fauna, the complexity of the natural world. To the right is order, the precision of geometry and the compass. In the middle, between these two incongruous elements, sits Man.

Except, in this case, it isn't just *any* man, it's Isaac Newton, writer of *Principia Mathematica,* the founding work of classical mechanics. Blake's painting is a criticism of Newton's world view; he is turning his back upon the beauty of the natural world, his sole interest is in his scroll and compass.

William Blake was a Romantic, with a capital R. He was part of the Romanticism movement, the artistic, literary, and intellectual reaction to The Enlightenment of the late 18th century. The Age of Enlightenment was the time that heralded the birth of modern science, when purest reason

Figure i.9

Newton, by William Blake is a beautiful illustration of the incongruity between the chaos of nature and the precision of science and mathematics.

William Blake (1757–1827) *Newton*, ca. 1795, print with ink and watercolor, Tate, London

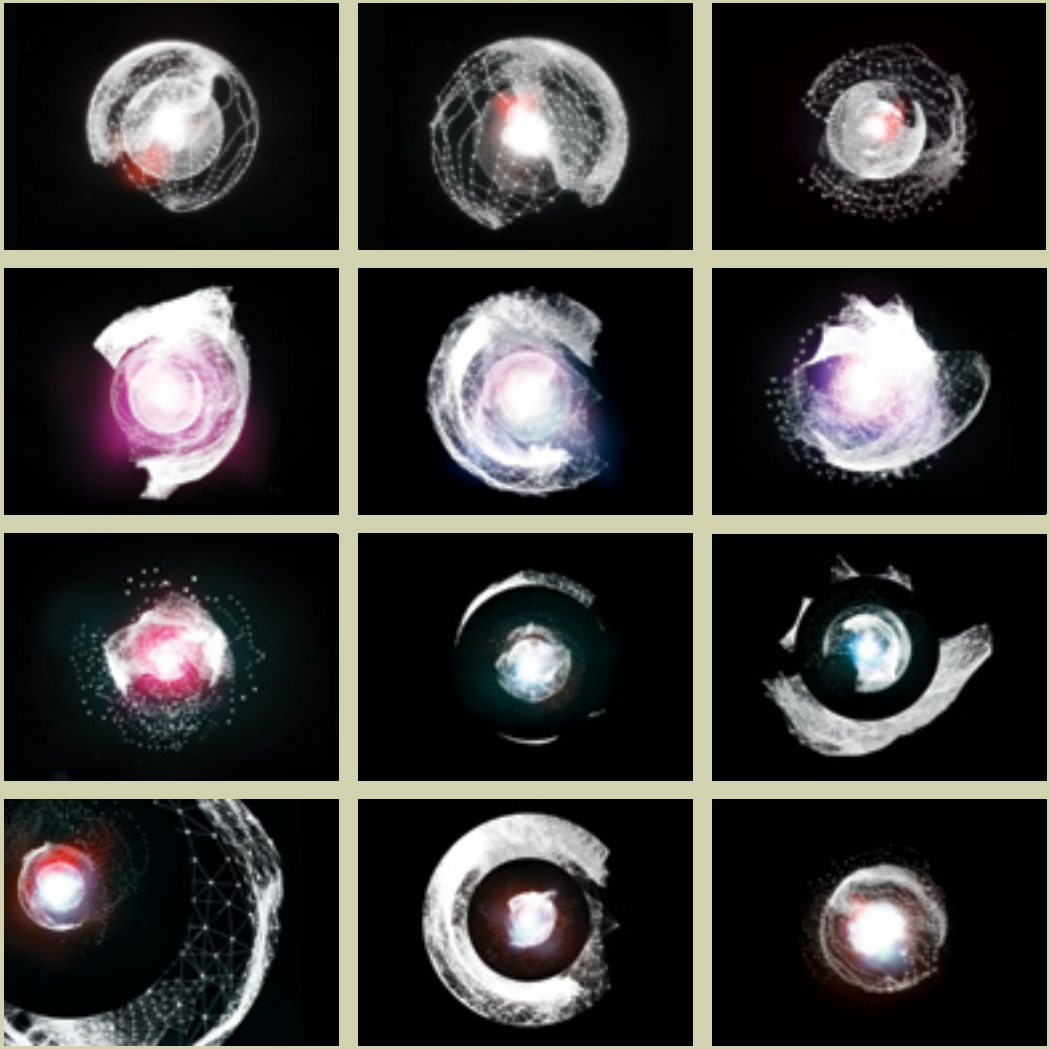


Figure i.10

Life in 2050 (2010), a video ident I created for a film festival, again collaboration with FutureDeluxe (http://futuredeluxe.co.uk). It was made using Perlin noise and deconstructed spheres, a process described in chapter 5. The finished film is at http://vimeo.com/10924639.

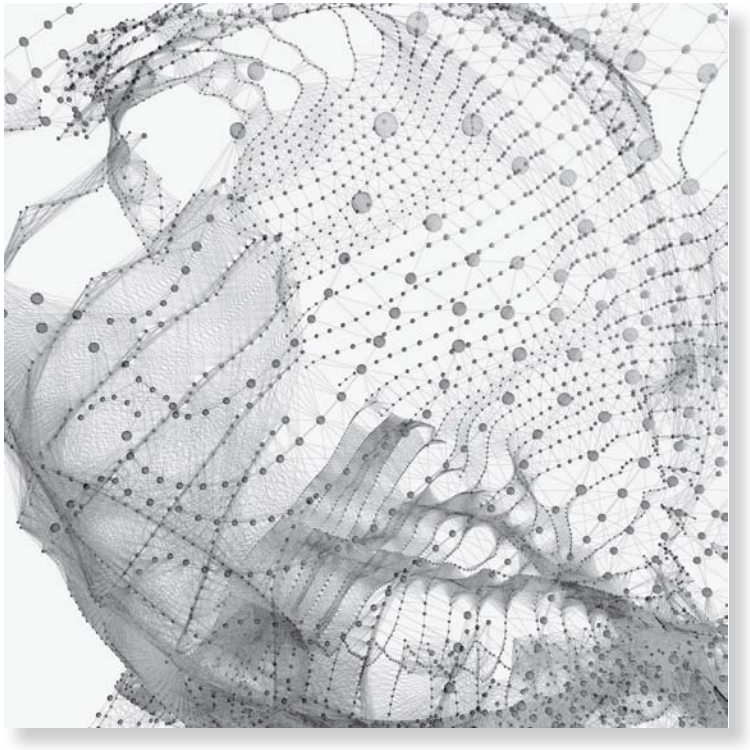


Figure i.11

Frosti (2010). Here, I reused the same system that generated the video work in figure i.10 for a print work.

was the dominant philosophical trend. The Romantics feared the coming godless world and clung to the dying remnants of an idea of natural idyll: the aesthetic, the rural, and the picturesque. They saw the future on their horizon, the world of the rational and scientific, the future we now live in. And it repulsed them.

Modern computer programmers may adopt the same pose as Newton. They spend their working days entranced before a screen, squinting at a glowing monitor in a dimmed office, making only the barest micro-movements with their mouse-hand and keyboard fingers, only vaguely aware of what is beyond the screen, outside the window, outside the city. The chaos of the natural world isn't welcome in this world of logic; unpredictability isn't something we want from a computer.

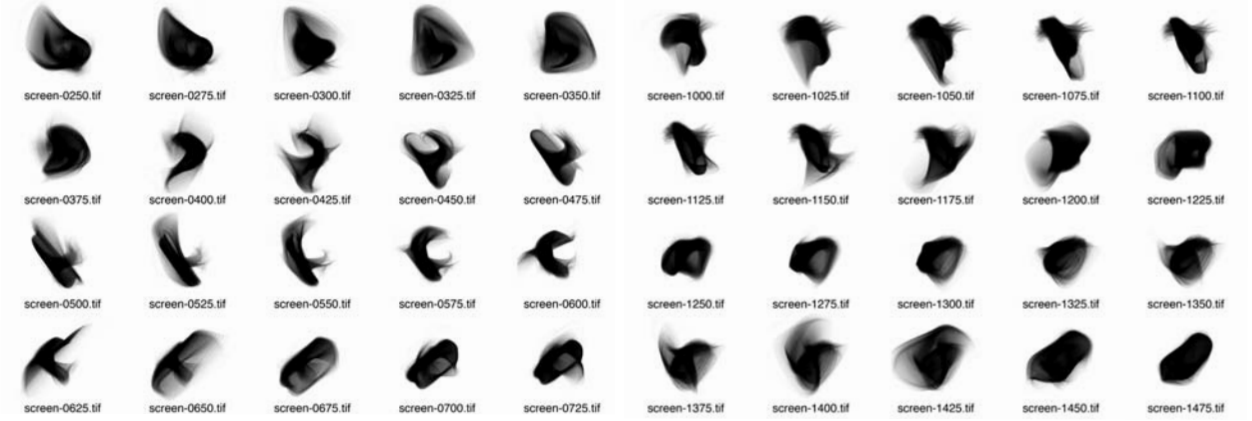


Figure i.12

Contact sheets for *Twill* (2010). This book's cover image was chosen from a run of several hundred generative images.

But the generative artist would be the programmer sitting facing the other direction, straddling that rock like Christine Keeler.[1] The generative artist comes from the world of logic to look toward the natural world for inspiration.

Is there an inherent contradiction in using computers to explore the realm of the organic? What could be less mechanical than the 1s and 0s of a humming box of electronics? Is generative art little more than a fool's errand?

And furthermore, if we declare our work as art, aren't we implying an intent to explore *emotion* and *aesthetic*? Can we reasonably expect to create works with emotional resonance using only procedures, logic, and mathematics? Relying on a computer to create an artwork and expecting it to connect with a viewer on a human level is surely akin to waiting for an infinite number of monkeys, with an infinite number of synthesizers, to write the perfect pop song.

Herein lies the challenge.

Order and chaos, simplicity and complexity, the mechanical and the organic, aren't necessarily at opposite ends of a spectrum. They're symbiotic, intertwined. Any line we might walk between the two is a knife edge. Our very existence is poised between entropy and order, between the

1. If you aren't old (or British) enough to remember the scandalous tale of John Profumo and Christine Keeler, visit Wikipedia to learn what a "defiant straddle" looks like: http://en.wikipedia.org/wiki/Profumo_Affair.

Figure i.13

Process 14 / Image 4 by C. E. B. Reas (2008). Casey Reas was one of the initiators of the Processing project. This print is from a series of Processing works exhibited at the bitforms gallery in New York in Spring 2008.

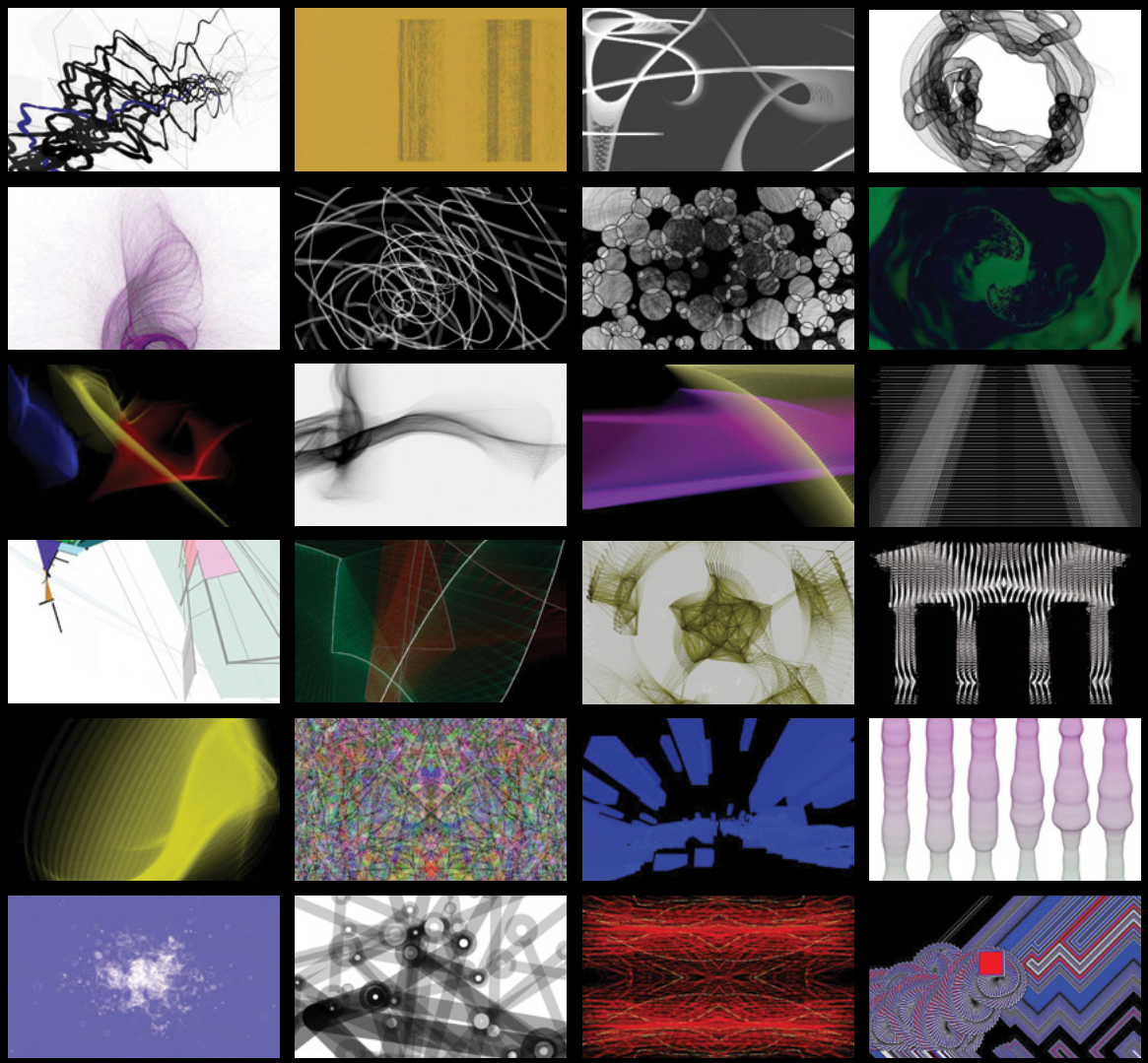
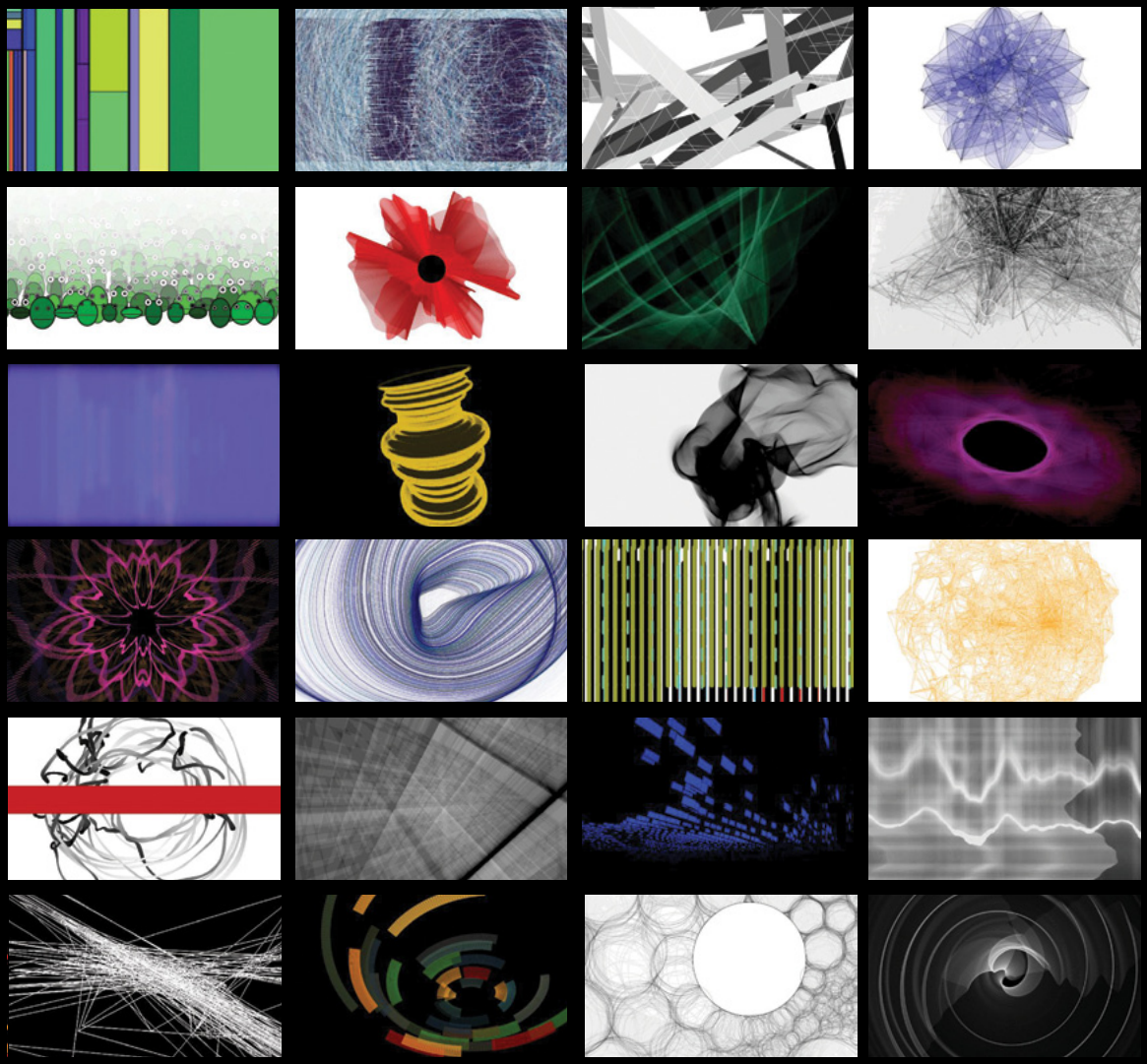


Figure i.14

100 Abandoned Artworks (2008–2010). Two years of weekly generative animation experiments, and their source code, which you can find at http://abandonedart.org. The code for many of the other works in this book can also be found there.



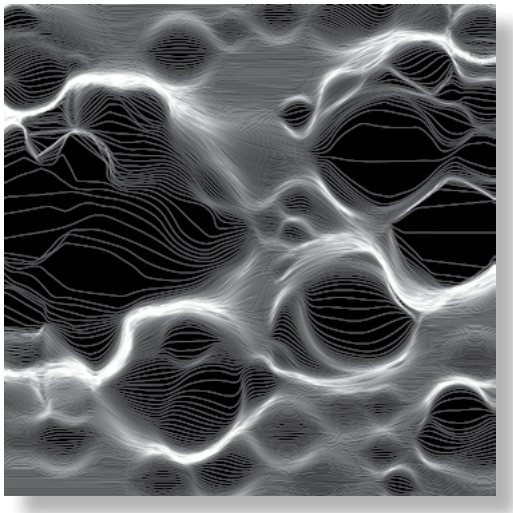


Figure i.15

(Left) Grid Distortion 02D 0018, a laser etching by Marius Watz (2010)

Figure i.16

(Right) KBGD01E 0012 by Marius Watz (2010)

turbulence of a hostile, chaotic environment, the natural world that we would find so difficult to survive unaided, and the simplicity of purest nothingness, the void that is equally fatal to our animal needs. We may think we have both mastered the chaos and fought off the boredom and madness of purest order. But any living being can testify that there is a constant fluctuation between the two. No matter how much order you impose upon your life, chaos is never more than a car crash away.

The mechanical and the organic, like order and chaos, are codependent; one couldn't exist without the other. The complex appeals to us as much as the simple, the organic as much as the mechanical. Fashion or mood may sway us more toward one or another in any given situation, but we never go completely over to one side or the other. For to do so is to stop living; to eradicate chaos is to become a robot; to eradicate order is to become a savage.

Fortunately, no one is asking us to take sides. The aim of generative art, if it has any aim at all, is to make something beautiful. We can attempt to use the mechanical to create the organic, starting from order and heading toward chaos, careful not to stray too far in one direction or the other. And if we become adept at this, we may be able to consider ourselves both programmer *and* artist.

Figure i.17

Illuminations B by Marius Watz (2007). Marius, who provided this book's foreword, is another generative artist whose work has been a great inspiration to me. Two more of his creations appear in figures i.15 and i.16. You can explore his other work and processes at www.unlekker.net.

Figure i.18

Renders from Aaron Koblin's *Flight Patterns* (2005), a data-driven work that is discussed in chapter 7

Programming as poetry

Some may think there is something counterintuitive about using a programming language as an artistic tool, like using a forklift to perform a ballet or a T-square to draw a curve. This is because, traditionally, programming languages have been the realm of logic, structure, and organization. They're used for problem solving, modeling data, and accurate simulations. Highly logical activities such as these inevitably make coding unattractive to more creative thinkers. This is a waste. Programming languages are just tools; they don't belong to one community or another. They can be equally effective in the hands of a designer or in the hands of a systems architect, but the works created by these tool users will be radically different.

The history of programming has been toward ever-greater sophistication of organization. When computer scientists talk about design patterns, they don't mean aesthetics; in programming lingo, a design pattern is a useful framework for tackling programming problems homogeneously. The intention is that common challenges can be addressed the same way, in a move toward a more universal language of programming. The use and understanding of these tried and tested design patterns is typically the mark of a good programmer, but they're also a highly rigid way of

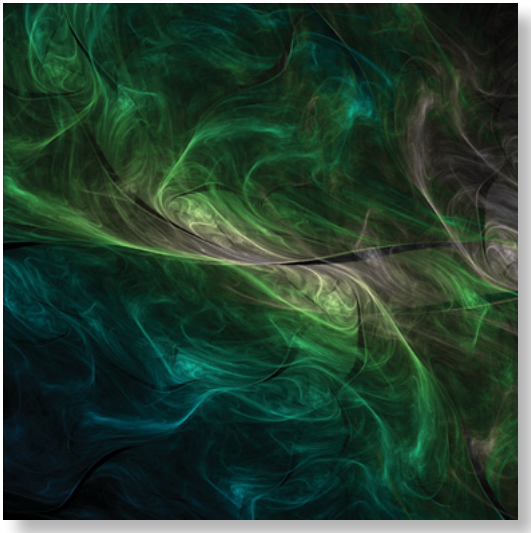


Figure i.19

Strange Symmetry 2 by Frederik Vanhoutte (2008)

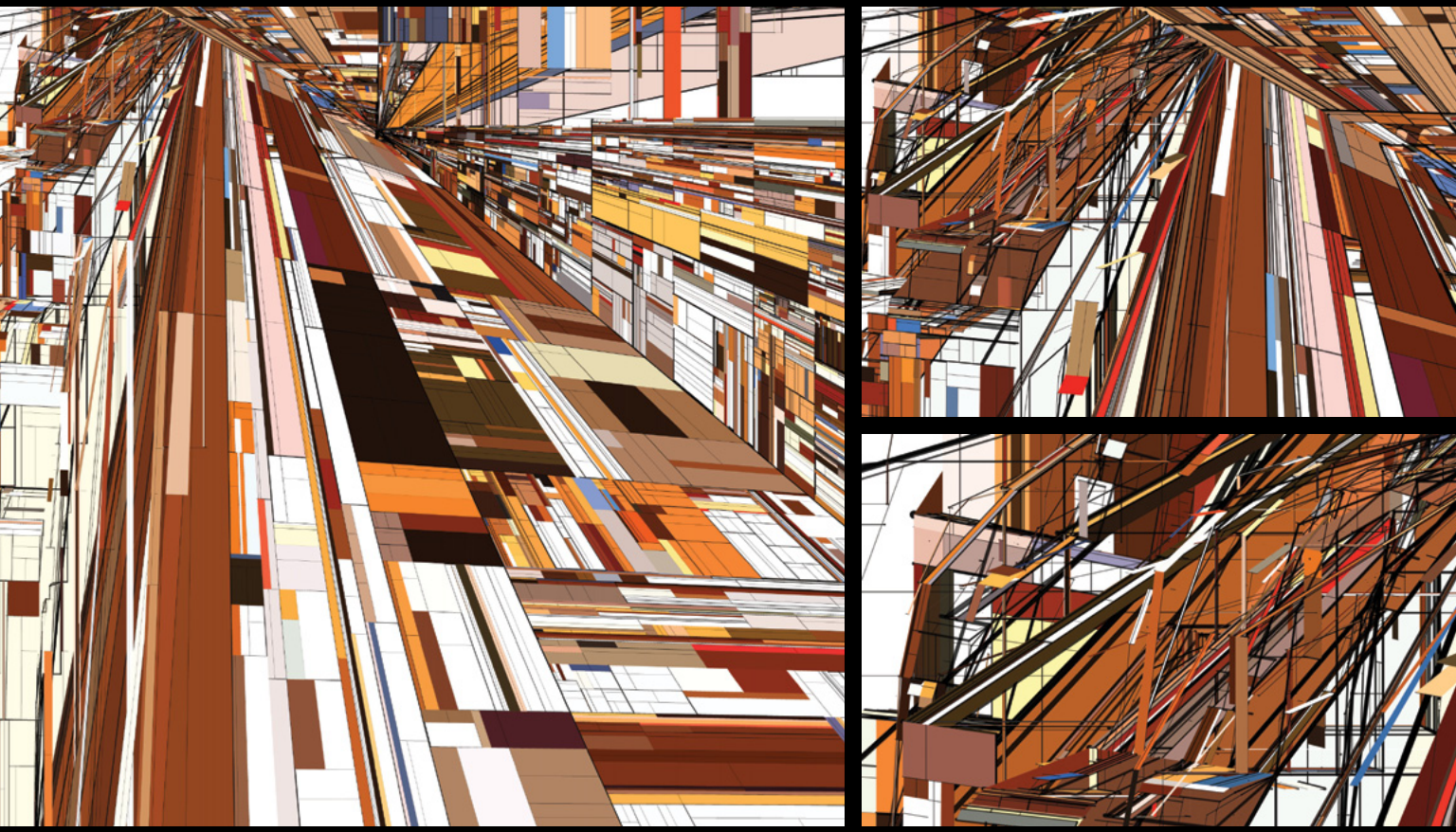


Figure i.20

Mondrian Architecture (2009). A fractal inspired by Piet Mondrian. The joy of creating fractal structures is exploring the detail: they can truly be said to work on many levels. Each detail is from zooming in to the upper-left corner of the previous image. These ideas are explained in chapter 8.

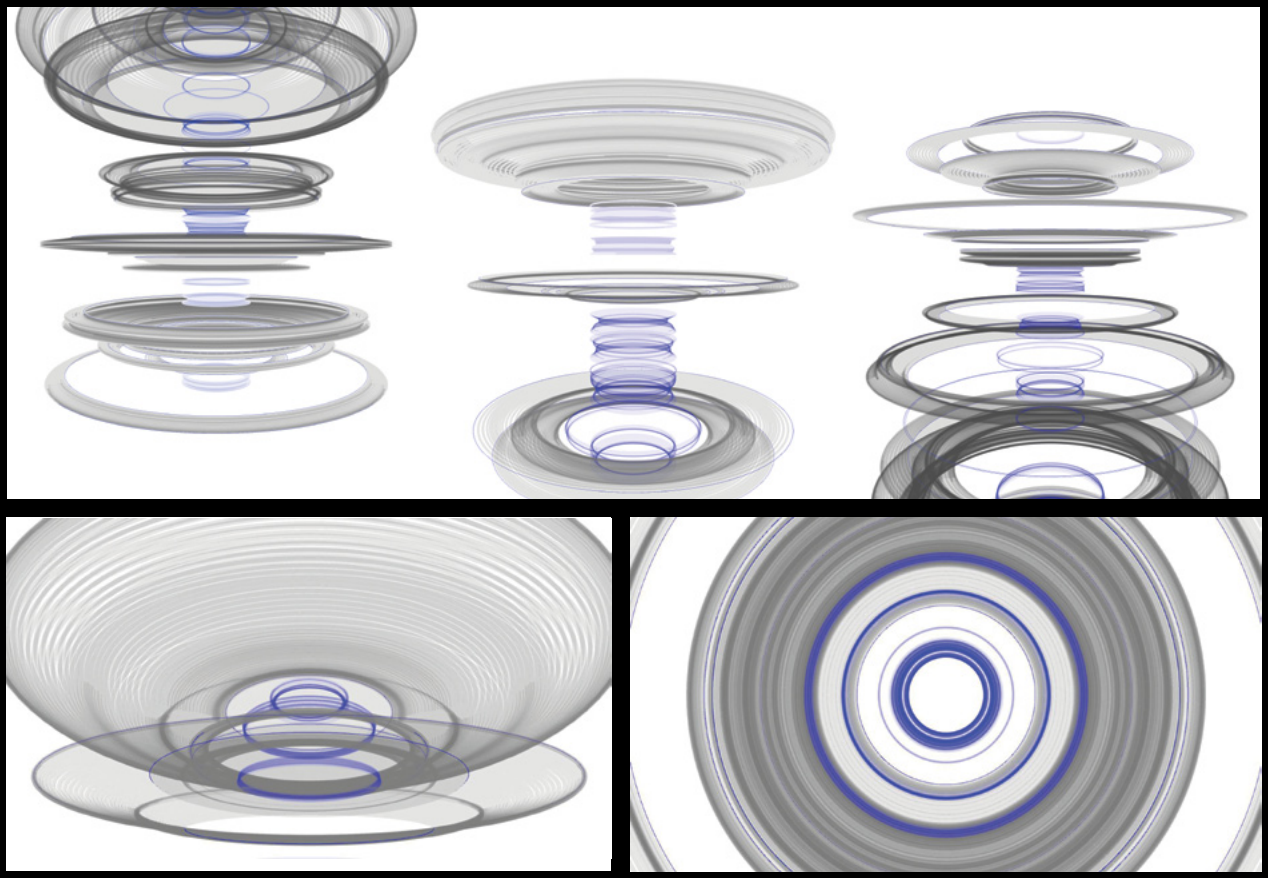


Figure i.21

Orbitals (2009). Perlin noise applied to 3D circles. All the necessary principles behind this work are discussed in chapters 4 and 5.

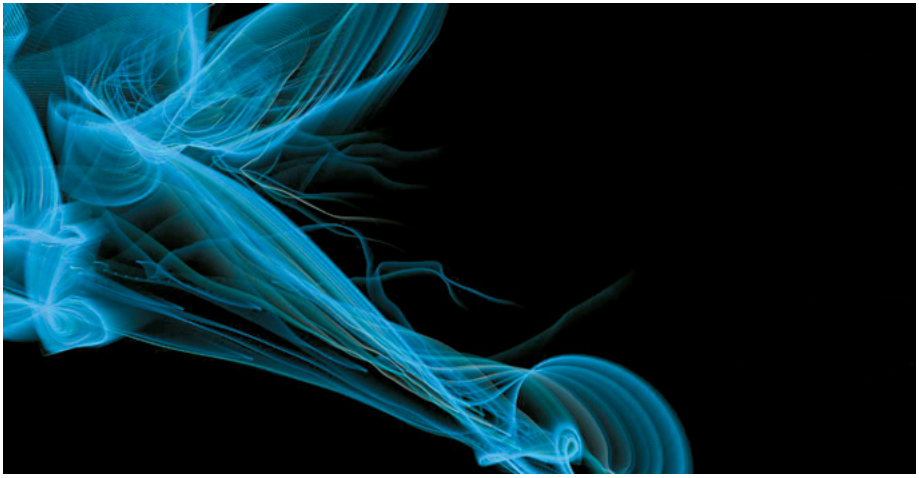


Figure i.22

Broken Mirrors 1–4 (2011) Still from a gallery installation I built. It shows one of four generative animations created from the movement of the spectators viewing the piece

approaching language use. What we're doing with generative art is swimming against this tide and saying there isn't a right or wrong way to do anything. We can still use these fine practices, and we can learn a lot from them, but we shouldn't allow ourselves to be hindered by them. If rules exist, they're for the breaking.

But not the rules of programming syntax, unfortunately. It doesn't take long to discover that there's nothing iconoclastic about missing a semicolon: you just end up with broken code. This is an unfortunate reality.

A programming language is, after all, just another language. And a language can be spoken in many different ways, with a variety of accents or inflections. The programming language differs from languages such as English, German, and French only in that it's intended to facilitate communication between humans and machines, rather than humans and other humans. Because of this necessary unambiguity of human-to-machine communication, we wouldn't imagine that we could use a language like Java or C++ to write a poem. But if a language isn't capable of poetry, it has clearly lost its relevance on the human side of the equation.

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

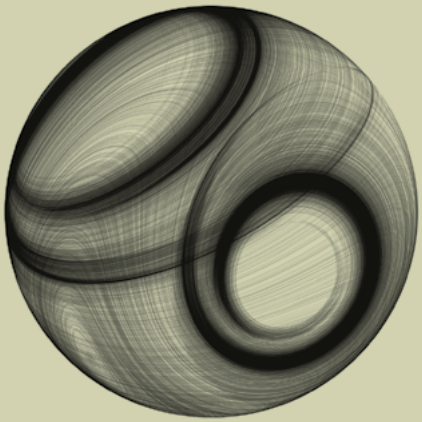


Figure i.23

LORMALIZED by Reza Ali (2010). Ali published this work as an app, so others could create their own images using his system. You can download it from www.syedrezaali.com/.

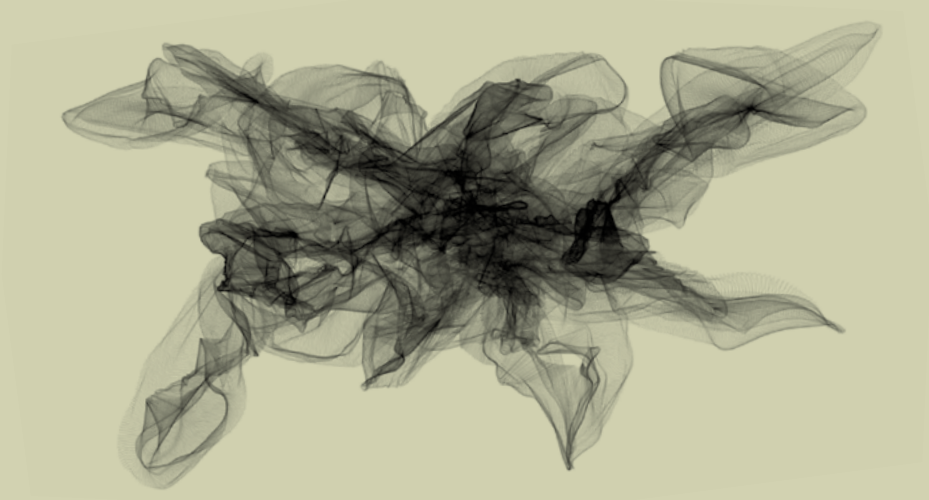


Figure 1.24

Drawing With Particles by 01010101 (2009). Jerome Saint-Clair, aka 01010101, fashioned this image with moves of the mouse, but the tool his mouse controlled was a generative particle system of 01010101's own creation.

Figure i.25

Happy Place by Jared Tarbell (2006). Agent-oriented behaviors at the micro level, producing emergent complexity at the macro level. If these words mean nothing to you right now, chapters 6 and 7 will explain.

The chaos artist

Declaring our work as “art” is a bold and arrogant thing to do. By doing so, we’re saying that our work is beyond mere utility: it’s an expression of our humanity and individuality. It may even be in the realm of the ineffable. Unfortunately, it may also be an expression of our pretentiousness, but let’s try not to worry about that for now. If you’re of the opinion that nothing in life is worth doing if it’s without specific purpose or measurable value, you’ve picked up the wrong book.

Furthermore, to describe our work as “generative” art means we not only intend our work to express our individuality, but we also want it to express the chaos and abandonment of processes free of our control. Without this unpredictability, our art would belong in a box other than the one labeled *generative*. Generative artists are chaos artists. They have bred the unpredictable, welcomed it, harnessed it, and can fashion it into pleasing forms.

Common or garden chaos (as opposed to the mathematical kind, which we’ll touch on in part 3 of this book) may seem anathema to the logician. Unpredictability is something unwanted in traditional programming practices, which is why more experienced programmers reading these words may be finding some of these ideas unpalatable. But trust me: it’s for your own good. Nonlinear approaches to the familiar can be healthy for the brain. And, who knows, you may even be able to take some of these creative angles back to your coder day job and make yourself incredibly unpopular with your bosses and colleagues.

Chaos isn’t to be feared. After all, we aren’t doing anything more dangerous than making attractive imagery with computers. Seriously, what is the worst that can happen? To engage in generative processes means we are *expecting* the unpredictable—it isn’t an unwelcome visitor. We’ll become comfortable with a lack of control over our work. We’ll embrace this chaos and learn to love it. ■

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