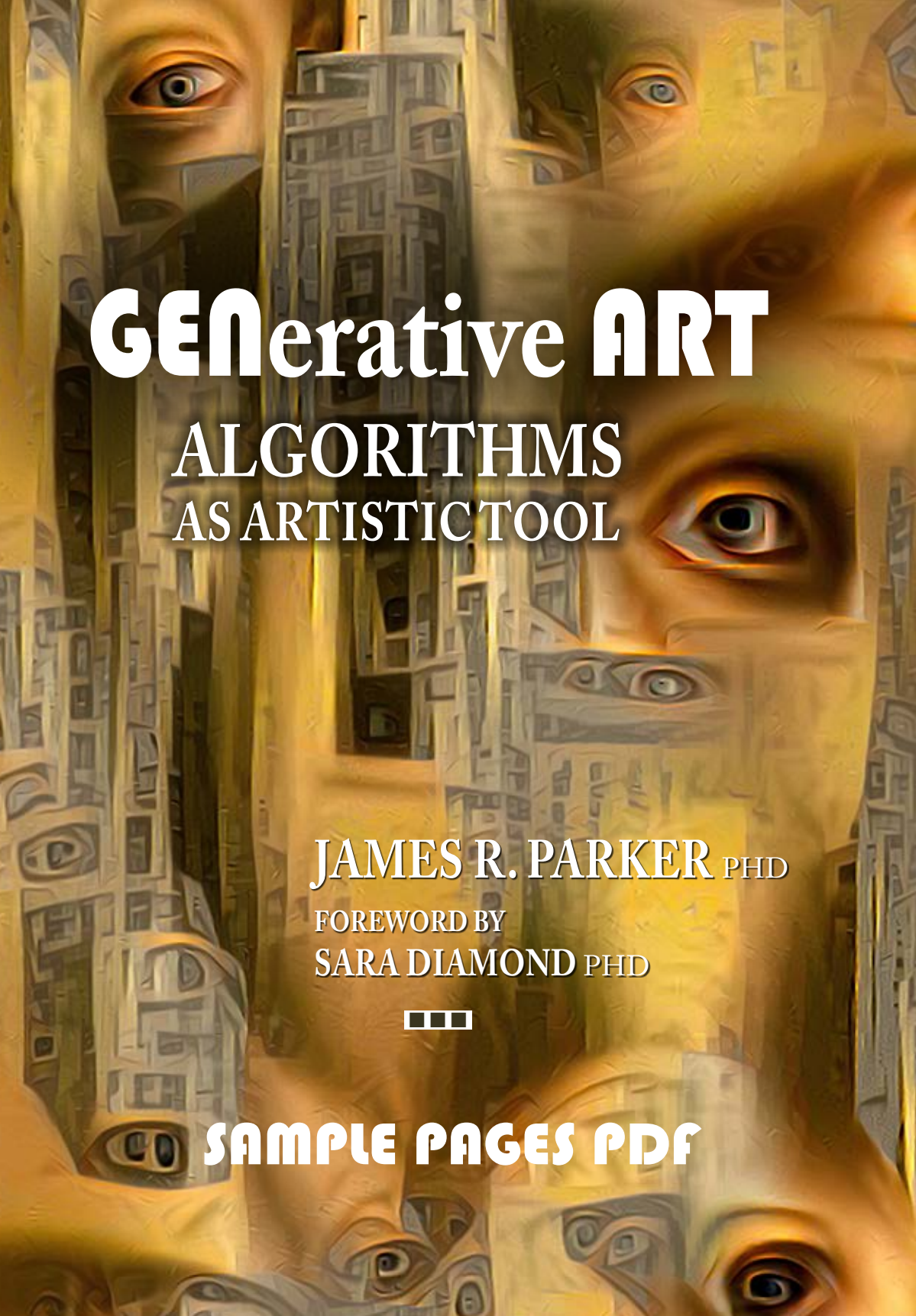

GENerative ART
ALGORITHMS
AS ARTISTIC TOOL
JAMES R. PARKER PHD
FOREWORD BY
SARA DIAMOND PHD
SAMPLE PAGES PDF

GENerative ART

SAMPLE PAGES PDF

Also by James R. Parker

Practical Computer Vision Using C

Algorithms for Image Processing and Computer Vision

Start Your Engines

Guide to Computer Simulations and Games

Algorithms for Image Processing and
Computer Vision (2nd Edition)

Game Development Using Processing

Python: An Introduction to Programming

Python 3 Pocket primer

RumbleSat: Art from the Edge of Space, *ed.*

Game Development Using Python

GENerative ART

ALGORITHMS
AS ARTISTIC TOOL

James R. Parker

With a Foreword by

Sara Diamond

SAMPLE PAGES PDF

DURVILE &
UpRoute Books

Calgary, Canada

UPROUTE IMPRINT
OF DURVILE PUBLICATIONS LTD.

Calgary, Alberta, Canada
www.durvile.com

LIBRARY AND ARCHIVES CATALOGUING IN PUBLICATIONS DATA

Generative Art: Algorithms as Artistic Tool
Parker, James R.; Author
Diamond, Sara L., Foreword

1. Art and Computers | 2. Algorithms
3. Computers | 4. Canadian Art

Book Six in the UpRoute Art & Artists Series
Information on this series at durvile.com.

ISBN: 978-1-9888243-8-3 (print pbk)

Cover art "The City Ponders" by Steve DiPaola with Graeme McCaig
Book and jacket design by Lorene Shyba

We would like to acknowledge the financial support of
the Government of Canada through Canadian Heritage Canada Book Fund and the
Government of Alberta through the Alberta Media Fund.

Canada Alberta Government

Durvile is a member of the Book Publishers Association of Alberta (BPAA)
and Association of Canadian Publishers (ACP).

Printed in Canada

First edition, first printing. 2020

For My Mother

She was a typesetter and lost her job to computers.

She does not blame me.

SAMPLE PAGES PDF

TABLE OF CONTENTS

Foreword, Sara Diamond

Preface

Introduction

Chapter 1 Generative Art and Computers

Chapter 2 Visual Coding

Chapter 3 Randomness

Chapter 4 Curves

Chapter 5 Images

Chapter 6 Animation

Chapter 7 Interface

Chapter 8 Net Art and Web Art

Chapter 9 Three Dimensions

Chapter 10 Down the Rabbit Hole

Sample gallery pages

About the author and cover artist

FOREWORD

Drawing a Line in Silicon

Sara Diamond

Sara Diamond PhD *is President of OCAD University in Toronto, Ontario. She writes and lectures on the subjects of digital media history, visual analytics, and mobility and design strategy. Her artwork is held by prestigious collections such as the Museum of Modern Art, NYC and the National Gallery of Canada.*

Generative Art: Algorithms as Artistic Tool combines three approaches to generative art. It shares philosophical, historical, and conceptual knowledge of code-based art-making; it offers 'how-to' staging that enables learners to learn and apply Processing and related tools; and it serves up a set of 'sketches' that provide exercises, code samples, and demonstrations of the author's works. Written in a personable and accessible style that reflects James R. Parker's firm foundations in computer science and art-making, this book is a unique new addition to textbooks that provide both concepts and tools to emerging computer-based artists.

Whence/Hence Generative Art

In the twenty-first century generative art encompasses a vast and expanding field, applied to any practice in which a computer program is coded by an artist to create experiences, and where the interplay of human intelligence and machine response drives the results. Applications vary from experimental visual art to animations, to augmented and virtual reality, to game design, to interactive installations. Author James R. Parker notes that the experience of the viewer, listener or interactor can be interactive, dynamic or passive.[1] Mark Guglielmetti, and Mark McCormick propose,

> The underlying principle in generative art is to create a process that generates an artwork from the process. A process might be a set of rules or a series of instructions that are carried out by others or by a machine. Each rule or instruction may be written in everyday language and is usually straightforward and unambiguous to understand.[2]

Generative art draws from a history before machine-based computation, rooted in mathematics and the creation of rules-based expressions of patterns by humans. The first algorithms were invented by the Persian scholar Muhammad Khwārizmī [3] in the ninth century and created visual representations that remind us of computer graphics today. Islamic art forms such as arabesques and tessellations are generated from repeated geometries. Ira Greenberg observes that, "Weavings, textiles, engravings, mandalas, and graphs from cultures around the world employ complex algo-

rithmic patterns based upon mathematical principles, yet most often are not developed by mathematicians."[4]

Generative Art: Algorithms as Artistic Tool chronicles the examples of Paul Klee and Fred Whipple, who created powerful images predicated on mathematical principles and were, apparently, generated randomly. Geometries, algorithms and patterns were fundamental to the development of expressionism and early painterly abstraction in the West as Stephen Luecking demonstrates, "In the development of abstract art geometric objects and patterning offered a ready-made and familiar category of abstract objects to which artists could refer."[5] Wassily Kandinsky, the Bauhaus teacher and abstract artist, was renowned for his eloquent expressions of geometry and form. Russian artist and designer Sonya Delaunay founded the Orphism movement in the 1910s, which used process and geometry based methods to create images that appeared to collapse different experiences of time and moments in time, capturing the power of the industrial revolution. These practices were also described as,

> …*simultanéisme*. Simultaneous design occurs when one design, when placed next to another, affects both; this is similar to the theory of colours (Pointillism, as used by Georges Seurat) in which primary color dots placed next to each other are "mixed" by the eye and affect each other. [6]

Parker provides techniques which explore the pixel (the color dot) as a fundamental building block of generative art, drawing from the techniques of the expressionist movement and artists such as Delaunay.

Drawing a Line in the Sand on AI

Generative Art: Algorithms as Artistic Tool arrives at a perfect moment in human history as the fourth industrial revolution charts new emerging relationships between humans and technologies. In an era in which Artificial Intelligence is positioned as both existential threat and transformational opportunity this book threads a careful course, underscoring the potential of a "generative art" that uses computational tools to augment and express human imagination without supplanting human expression and experimentation. As Fred Popper notes about art work which both applies and invents technology,

> …technology is or can be humanized through art. It is true that something exciting happens when one looks at a familiar subject not as a closed conceptual system but to find an ongoing conceptual edge—in this case, the increasing humanization of technological virtualism.[7]

This is the promise of generative art. Parker asserts,

> Generative art uses a computer as an enabling technology. The work is generated by a computer program, but is not fully autonomous. The computer is guided to create the work using detailed instructions from the artist, who either writes the program or at least creates the method (algorithm) for the creation of the work.[8]

The author underscores the importance of perspective in the creation of the art work, and these

perspectives include most importantly that of the creator and the culture that they bear as well as the technologies that they produce. Rather than pessimism, the creative tools of generative art and the associated methodologies of coding give artists the ability to transcend medium and integrate emerging new media into their practices. As bioinformatics art and science artist Dr. Victoria Vesna proposes, when artists understand and share the tools of science, new opportunities emerge for creative practices and discovery,

> The relationship between art and science has always been important but now that we use the same computing tools, we are able to work together, exchange ideas and create something in between that is about creative, innovative thinking. Scientific innovations change our reality and artists have always been agents of change and portraying reality in new ways, pointing to new visions, potential dangers.[9]

Generative art can leap boundaries and frontiers.

The Conceptual Leap for Artists: Learning to Code

For many decades artists working in "new" and now digital media have debated whether or not learning how to code, versus using pre-packaged creative tools, or collaborating with programmers to develop complex art and technology creative works. While these are different models of practice, in the 21st century, where coding, and technology driven decision-mak-

ing are driving all fields of endeavor, it is imperative that artists learn how to code. Even if artists do not ultimately do the programming they must understand the capacities and limits of software. The conflation between artist and programmer increasingly appears in programs at universities around the world, as well as festivals, science conferences and exhibitions such as the International Symposium of Experimental Art (ISEA) a two-decade-long showcase of artists who work with algorithms. Its 2019 event in Gwangju, 'City of Light', applied creative computational tools to 'Umbra and Penumbra', 'light and sound', 'Data Analysis', 'Data Visualization' with the goal of extracting meaningful signals from noise, and harmonization'.[10] There are many more examples. As Parker (2020), Reas & Fry (2007) and Greenberg (2007) note, Processing also works for artists who approach computation through the moving image – it is extensible and effectively supports the inclusion of software based animation tools.

Generative Art: Algorithms as Artistic Tool guides artists in their programming learning journey. Computer code is a descriptive language, in which symbols are used to communicate, requiring the coder to conceptualize abstractions, and approach images in both linguistic and mathematical ways. For artists, one challenge is to translate their learned art-making approaches into a language that a machine can interpret, or as Veronika Tzankova and Michael Filimowicz note, "Code is a text but of a specific kind – a text that

produces causal chains in machines rather than intertextual effects of encultured associations."[11] Computers demand precise description in which the artist must build an imaginary structure in their mind that they translate algorithmically, describing images in mathematics. According to Parker,

> A sketch of the solution, perhaps on paper in a human language and math, is created. This is translated into computer language and then typed into the computer form using a keyboard. The resulting text file is called a script, source code, or more commonly the computer program.[12]

Casey Reas and Ben Fry, the inventors of Processing, created a program that would build on known programming methods but allow creative expression. At the same time, they recognized the constraints that computation imposes,

> Each computer program reveals the style of its author, but there is far less room for ambiguity. While people can interpret vague meanings and can usually ignore poor grammar, computers cannot.[13]

Parker guides the user/reader through learning precision in coding, while also learning how to ensure that the ambiguity of the human touch is rendered within creative computational practice.

Without question, individual artists approach coding in different ways. Ira Greenberg, writing about his adoption of Processing, describes his intuitive coding practices, a "steal, tweak, pray coding approach," which

matured into systematic planning and learning of programs that in turn allowed intuitive applications.[14] Parker leads his readers/learners through an accelerating path that brings together concepts, methods, exercises and examples. Chapters help the user/reader bring their extant visual knowledge to the fore while introducing algorithmic logic. We learn that realism ironically requires randomness, as humans are inconsistent in their repetitions, with tools such as thick line code, and that realism benefits from inclusion of texture to invoke sensory experience. We discover the pixel as building block, and that the image is, "a two-dimensional grid of pixels that gives the impression to a human of being a coherent picture."[15] We are asked to fathom experiences of time, from Absolute, to Relative time, to Duration, in order to comprehend time within animation. While deciphering these complex concepts *Generative Art: Algorithms as Artistic Tool* always returns to the pragmatics of making.

The Book as an Interface

If we think of *Generative Art: Algorithms as Artistic Tool* an interface, it effectively echoes the ways that working with code plays out. Inventor Alan Kay was adamant that human cognition must combine, "a doing mentality, an image mentality, and a symbolic mentality," the latter to, "tie together long chains of reasoning, [to] abstract."[16] *Generative Art: Algorithms as Artistic Tool* will provide a new generation with the pragmatic tools and underlying understandings of algorithm-driven generative art practices. ■

Notes to Foreword

1 Parker. James R. (2020) *Generative Art: Algorithms as Artistic Tool.* Calgary: Durvile. pg 27.
2 Guglielmetti, Mark & McCormick, Mark (2017) "Between Code and Culture", In Filimowicz, Michael & Tzankova, Veronika (2017) Eds., *Teaching Computational Creativity.* Cambridge, UK: Cambridge University Press. pg. 200.
3 https://en.wikipedia.org/wiki/Muhammad_ibn_Musa_al-Khwarizmi
4 Greenberg, Ira (2007) *Processing: Creative Coding and Computational Art.* New York: Springer-Verlag, Friends of Designer to Designer. Pg. 5.
5 Luecking, Stephen (2013) "Mathematics Education and Early Abstract Art", In Proceedings of Bridges 2013: Mathematics, Music, Art, Architecture, Culture. Pg. 35.
6 Baron, Stanley; Damase, Jacques (1995). *Sonia Delaunay: the life of an artist.* London: Thames & Hudson.
7 Popper, Fred (2007) *From Technological to Virtual Art.* Cambridge, MA: MIT Press. pg. 5 .
8 Parker. James R. (2020) *Generative Art: Algorithms as Artistic Tool.* Calgary: Durvile. pg 19.
9 http://digicult.it/design/victoria-vesnas-interactive-experience/ Accessed May 20, 2019.
10 http://isea2019.isea-international.org/ Accessed May 20, 2019.
11 Tzankova, V., Filimowicz, M. (2017) Introduction: Pedagogies at the Intersection of Disciplines in Filimowicz, Michael & Tzankova, Veronika (2017) Eds., *Teaching Computational Creativity*. Cambridge, UK: Cambridge University Press. Pg. 9.
12 Parker. James R. (2020) *Generative Art: Algorithms as Artistic Tool.* Calgary: Durvile, Artists and Art Series. pg 32.
13 Reas, Casey & Fry, Ben (2007) *Processing: A Programming Book for Visual Designers and Artists.* Cambridge, MA: MIT Press. Pg. 17.
14 Greenberg, Ira (2007) *Processing: Creative Coding and Computational Art.* New York: Springer-Verlag, Friends of Designer to Designer. Pg. 29.
15 Parker. James R. (2020) *Generative Art: Algorithms as Artistic Tool.* Calgary: Durvile. pg. 120.
16 Kay, Alan (1989) "User Interface: A Personal View", In Packer, Randall & Jordon, Ken (Eds). 2001, *Multimedia: From Wagner to Virtual Reality.* London and NY: W.W. Norton and Company. pg. 138.

PREFACE

No Fun? You're Doing it Wrong.
James R. Parker

Many art books are called that because they contain beautiful glossy color images of amazing artworks. This is not one of those books. Some art books are called that because they are written about or by well-known artists. They discuss the life, trials, and works of these incredible people. This is not one of those books. Some art books are called that because they contain the avant-garde, the postmodern, and the simply odd. This is not one of those books.

This is, in some sense, a textbook. The intention of the author is to show you, the reader, how to create art in a relatively new genre – generative art.

If you look up my name on the internet you will see that I am a professor of art. That's true, but I feel like a bit of a fraud, and the reason is that I had no training as an artist until I became a professor. I studied mathematics and computer science at university, but then I worked with artists of many kinds: musicians, actors, writers, playwrights, painters, and photographers. The new information to which I was exposed sunk in, and if I am an artist now it is by osmosis, not by training. I hope that's okay.

I was known for work in image processing and vision in the 1990s as an academic. But then, in 1998, I registered for and attended the Game Developers Conference, and my life was changed. The energy there was incredible. People everywhere were completely enthralled by their work. They were having fun. They were doing things and speaking about things that I had not heard about in my academic venues, and they were fascinating things. Moreover, their work had an immediate impact on people. It was not about computers, but about what a designer and artist could do with them.

I began working with artists then. I was, to my great surprise and honor, allowed to be a part of the Prairie Collective, a group of artists in my city who were working on experimental performance and installation art. I was allowed to work on more traditional theatrical productions, like *Antigone*. I have worked on live online theatre, displayed my work in exhibitions, and even sold some. But I am still a programmer at heart.

This book arose from my experiences as an artist and my desire to return something to the discipline. My work as a professor of art has changed me and helped me. It has caused me to think differently about myself and the world. So, if a computer programmer can read this and have a similar experience then it will make me feel as if I have succeeded. If an artist reads this book and learns how to create works in a generative way then I have succeeded again.

I presume some willingness to do new things, but no previous programming experience. The mathematical knowledge needed to proceed through the book is low, high school math basically. I am keen to hear from people who had trouble with the math, and will post new material on the internet that I think may help. The same with the programming material. I teach a university class named 'Programming for Artists' and using Processing as the language, but I know that each person has their own personal difficulties with a subject such as this.

For the computer programmers reading this I have other matters to share. Art is not the same as visualization. A visual artist creates an image that has both aesthetic value and meaning. Using a computer does not alter that fact. Generative art is really the art of the algorithm, and so both artists and programmers have a problem to solve: how to describe accurately and in detail a visual form that has both appeal and significance. This work of art may include elements that have a random aspect.

Randomness is a key factor in the real world and in art. The world has many things that appear random, but which in fact are simply the combination of many interacting processes. The flow of traffic on a main artery is actually the result of a great many decisions made by individuals about what time to depart, where to travel, and what route to take. They each have distinct driving habits. The result is a random-seeming arrangement of vehicles on the road. One could simulate all of the details of this, or simply make it look random.

Human drawings and paintings have a similar property. A human cannot draw a perfectly straight line, but a computer has trouble doing anything but that. To give computer artwork a human quality, one must inject a degree of randomness into its creation. Lines can't be perfect, nor can color or shading otherwise they appear computer generated, even if they are, and hence emotionless, without the touch of the artist. Artists are directly involved in generative art, but it must appear as if that is so or sometimes the work will be rejected. A computer can't comment on human affairs, and can't represent human feelings.

Art is a fundamentally human exercise. Most of human activity involves art: we decorate our caves, we try to make our homes distinctive; we turn the simple action of eating into art. We sing and we dance, for no practical reason. Everything we do becomes art eventually.

And, of course, it's fun. If you are not having fun then you are doing it wrong. ■

• • •

About the Generative Art Web Site, Mink Hollow
Generative art is difficult to describe in a written book, but examples and program codes that are posted on the *Generative Art* website will be helpful. https://www.minkhollow.ca/parker/generative

—J.R. Parker on top of Big Hill, Alberta
December 2019

INTRODUCTION

The Random Nature of Art

"Chance must be recognized as a new stimulus to artistic creation."
—Dadaists Jean (Hans) Arp and others

Human beings create art in everything that we do. Consider food: we could run after wild animals, club them with a stick, and then consume them raw. In ages past, perhaps that's exactly what happened. We have tuned that basic function, that of eating, into art. It's hard to deny that Beef Wellington or *Crème caramel* are art. Consider housing: it is possible that humans once lived in caves, but even then we decorated them. Now we make elaborate structures decorated inside and out, with amazing colors and metal and glass and a huge number of aspects that are not really needed to house us. The same is true of clothing, transport, and every aspect of human life. We are the ultimate jackdaws.

There are words that are very hard to define with any precision, so when we think of art there could be many images in our minds. That's okay, and it introduces the concept of randomness in art at the highest level: not everyone thinks of art in the same way, and some artworks will have success and others not. It's not possible to design success into a work.

For something to be random it must be unpredictable. There are degrees of randomness as well. The weather is only partly predictable, for example. The exact time of the arrival of the letter carrier at your door is unpredictable, although it is probably known approximately. So it is with art. There are many aspects of art that are unpredictable to a greater or lesser extent, and that unpredictability is often due to the fundamental unpredictability of humans themselves. Humans are not predictable as individuals, and as art is a fundamental human activity, art is unpredictable too.

It is really not possible for a human to reproduce a drawing or painting precisely. We can get close, but precision eludes us, and is the domain of the machine. But that is only one aspect of randomness in the artistic realm. It seems to be a hierarchy of considerations, beginning with the most complex one – the thought processes of the artist.

CHAPTER ONE

Generative Art and Computers

"Art is, now, mainly a form of thinking."
—Susan Sontag

BEGINNING any discussion of a topic that is new to the audience most often starts from a definition. This can be a problem for something that is truly new or not well understood, or for things that do not have very good definitions. The word *fun* is one for which only poor definitions exist, as any computer game designer will tell you. The word *art* is one that is difficult to characterize. No single definition can truly convey what art is. How, then, can a discussion of generative art proceed from a position of no prior knowledge or experience? By example, analogy, and metaphor.

The first thing to understand about generative art is that it is not a movement or a style. A style would permit a positive assessment of a work as belonging or not to a particular class or genre on inspection. It would involve visual elements, for example, and characteristic image transformations that would allow works to be classified. Styles are often distinctive and recognizable, although in some instances the distinctions can be subtle. *Cubism*, for example, can be identified with some reliability, and even has some fairly clear properties that can be used to define it: the use of multiple views of a subject simultaneously in a single rendering, for instance.

Fauvism, as another example, is characterized by the fact that color is used as a compositional element independent of objects in the scene, at least independent of its representational role, and uses simple forms and saturated colors. However, generative art can't be identified easily by its colors, lines, shapes, or viewpoints. It is characterized largely by how it is made.

In some sense, generative art can be thought of as *constructivist*. Constructivist art is about how materials behave. How is plastic different from steel, and glass distinct from stone? In constructivism, the form taken by an artwork is defined by its materials, and not the reverse. An artist often takes materials like oil and tints and creates painted renderings,

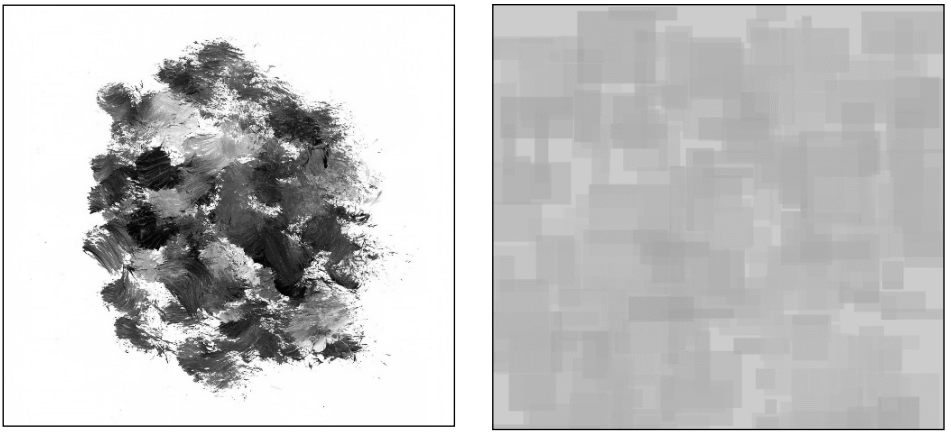


Figure 1.1 – Which is a painting?
Note: See images in color pages of this book.

transforming the raw material into a painting or drawing or sculpture.

Generative art uses a computer as an enabling technology. The work is generated by a computer program, but is not fully autonomous. The computer is guided to create the work using detailed instructions from the artist, who either writes the program or at least creates the method (*algorithm*) for the creation of the work. It is not the automatic creation of art by computer, but is a way for the artist to create meta-art. An artist will describe the work to be created as a product of their imagination, and then instruct the computer on how to build it. Thus, any style of art can be created. Generative art defines how.

Consider the works above. Which one is a painting and which is generative? The style of the work does not always betray the method by which it was constructed.

Generative art, according to some, is created at least partly by something called an autonomous system, which is some system that works on its own without human help. That's not quite true, because the human help occurred in the design of the work, while the 'autonomous' part occurs during the rendering process. A computer is in some sense an autonomous system, but is an elephant? If someone throws paint on a canvas from a great height are they in control of the result? Is that not some degree of autonomy? Moreover, is it not a matter of degree only – a sculptor may have pretty good control of a hammer and chisel, but it is not perfect. Painters used to make their own paints and brushes, but now they usually don't. Which gives them more control?

Some say that in generative art some of the decisions are made, not

by the artist, but by the system. That appears to be true always to some degree in any medium. If an artist relinquishes too much control over the work then there is a good argument to be made that the artist did not create the work at all. To quote Roger Ebert:

> I do not believe collaborative art cannot be art. I cite cathedrals and tribal dances as collaborative works of art. But they begin with an auteur with an original vision – whether that be a king, an architect, or a choreographer. The film director usually has the original vision.

Reducing an artist to an 'auteur' minimizes the skill required to paint or draw, compose or sculpt. On the other hand, perhaps that creative force underlying an artwork is the crucial aspect of that work. The opinion underlying this book is that *an artist should participate in both the conceptualization* ***and*** *the production of a work of art.* In the case of generative art, this means that the artist must be a programmer to some degree. So:

A generative artist is a computer programmer. At least he or she has some of the skills that a programmer has. This includes the ability to express their intentions in a computer language to at least some degree, to be able to communicate with other programmers using their particular patois, to be able to use typical computer interfaces, and to fathom the advantages and limitations of computer and digital representations for their art form. This places a generative artist in the same general category as a practicing physicist, archaeologist, sociologist, or generally any profession in the 21st century. All use computers, and all will be at the whim of some other programmer somewhere if all they do is operate predefined software tools like *Photoshop* or *Final Cut*.

Those are good tools but a user will always be limited to those specific operations that the implementer decided to implement. A programmer can bend the computer to their will, whatever that might be. An artist who is also a good programmer will never be limited in what visual presentations or messages they can convey. They can create entire universes from their imaginations and have others not only see them but also be able to walk around within them. In the 21st century, programming is a skill that *everyone* will find important.

Programming is not about mathematics. Math is a different tool, and an exceptionally valuable one too. It is true that computers represent everything manipulated by them as numbers, and this means that a programmer does need to be able to identify numerical representations for everyday things like sounds and colors. There is a clear difference between numbers and mathematics, though, and the level of math needed to be a

programmer is not as advanced as many people believe. It is true that some methods used by people who can use the computer effectively can be pretty advanced, but most are not. A knowledge of calculus, for example, is relatively rare even amongst programmers unless their University degree required it. It is uncommon to need such math while developing a typical program. Some algebra is very useful, and most high school graduates have some. More about that later but suffice to say that the level of proficiency in algebra required to be a programmer is minimal. One simply needs to understand the use of names in place of numbers and the generality of an algebraic expression.

Geometry, of course, is a very useful subject to know if you are an artist. It is the boundary between art and mathematics. Images contain objects related by angles and drawn as lines, circles, ellipses, and other geometric shapes. Some idea about how these can be manipulated is very useful indeed. Some trigonometry too, although again one really needs only the basics: the concept of distance, Cartesian (rectangular) coordinates, and the way to find a point given an angle and distance, and so on. No really odd 'identities' and certainly no proofs are needed. On the other hand, if mathematics does not frighten you (and why should it?) and you feel capable of learning more, a better knowledge of math will allow more advanced and interesting concepts to be explored.

Math seems to be about symbols and a formal method for manipulating them. In fact, it is about deeper relationships than that. The symbols are a language used by mathematicians to communicate ideas between themselves and permit them to demonstrate the correctness of their ideas. The importance of math is in its ability to accomplish useful things and conceptualize abstract ideas. Trigonometry, for one example, is concerned with relationships among triangles. Certain things have been observed over the many centuries that humans have been aware of our surroundings, and at some point in our history triangles became important. We discovered that heights of things could be found from the ground, using angles. This was important knowledge. Then again, the fact that *sine (2θ) = 2*sine(θ)*cosine (θ)* is not of frequent interest to most people except mathematicians. Also, why the Greek letters?

As an obvious example of mathematics being used in art, consider the case of fractals. A fractal has been described, by Mandelbrot, as "a rough or fragmented geometric shape that can be split into parts, each of which is (at least approximately) a reduced-size copy of the whole." It's a good definition but does not really lead to a clear insight into what a fractal might really look like. The simplest example is something called the *Koch snowflake*, which begins as an equilateral (all three sides equal in

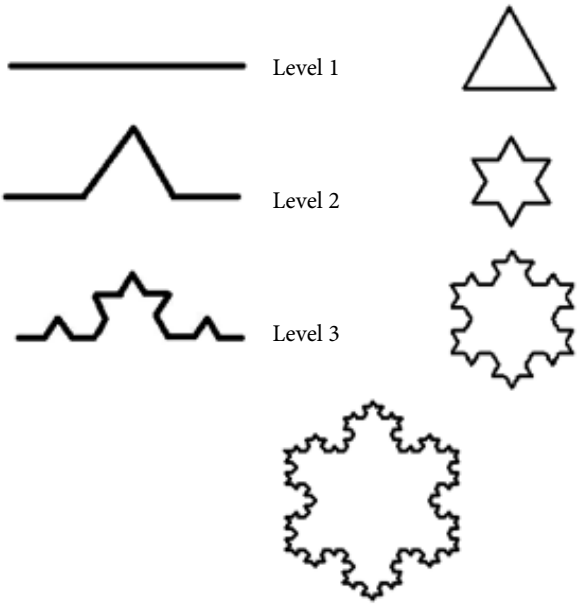


Figure 1.2 – Basic fractals

length) triangle. Now on each existing edge, add an equilateral triangle in the center with sides 1/3 as long as the original edge and pointing outwards. Now do it again, and again. Forever.

Each time the repetition occurs the result is a new level or degree of the figure. Higher levels are more complex than low ones in terms of the number of lines involved, but the final curve, after an infinite number of levels, has a finite area and an infinite perimeter. This is hard for most people to comprehend and is a part of the power of formal mathematics – it can represent and demonstrate things that can't be imagined.

So, like programming, mathematics is a tool that can be used in the creation of generative art. The more an artist knows about both, the better generative artist they will be. However, we don't have to know everything, and we'll learn more useful things with time and practice.

A generative artist has a visual concept and, rather than painting or drawing it, describes it to a computer. The description begins as a general idea and is probably expanded using sketches. It then must be converted into a sequence of operations that a computer can perform, a step-by-step description of the process leading to the creation of the conceived image. A computer scientist or engineer would call this an algorithm. The algorithm can be converted into a computer program, which is not always an obvious process, and the computer program will create the image when it executes.

The need to describe the artwork in great detail distinguishes a generative artist from others. A painter describes their work by painting it. The final artwork is the result of the generative artist's vision, and the description stage is equivalent to the painting or

drawing that another form of artist would do. Visual artists all start with a vision, and the next step labels them as one sort of artist or another.

A generative artist might not be able to draw or paint well but must be able to see their idea and to wield the digital tools of a desktop computer to render it.

Computers

Computers were originally built to do repetitive calculations with a high degree of reliability. Humans, you see, get bored and tired. Having a person do the same kind of arithmetic over and over again leads to fatigue and errors. The original solution was to have the same calculation performed by more than one person and then compare the answers. Correct answers tend to agree with each other, but mistakes are likely to be different. Mechanical calculators tend to either work or not, and it's pretty clear when they do not. Computers are the same way.

Because of their origin, computers have one essential property to keep in mind: **Computers can only manipulate numbers.** Of course, humans have written millions of programs that hide this fact. Computers can recognize voices and speech commands, can identify faces, can play chess, and do millions of things that appear to be non-numerical. The software hides what is happening underneath. To get a computer to create a picture or music, we must devise some sort of code that allows pictures and music to be made into numbers. Truly, at the lowest levels of abstraction, computers only handle numbers.

People see computers using letters and words, for example, every day. A keyboard is fundamental to a computer, and almost all computers have them. Surely, computers can manipulate letters. No, not unless a method is devised to convert the symbols we see as letters and the keys we press into numbers. Then the computer can manipulate the numbers. This is an encoding, and computers use many of them. Characters are just one of the first that was devised, and so it shall be the first one discussed.

A simple encoding: *characters*

Most people have had some experience with codes or ciphers, even if only as puzzles or kid's games. We can assign numbers to letters: 'A' is '1', 'B' is '2,' and so on. A message could be sent using these numbers, such as "8 5 12 12 15" would be "hello." This is an encoding, but not the one used by computers; it's too simple to be generally useful.

There are 96 printable characters (in English), and these are the ones that would appear on a typewriter. If we were building a numerical representation for characters for use on a computer, what considerations would be important? Of course, letters have both upper and lower case. Uppercase 'A' and lowercase 'a' would have a distinct number. We'd also have to decide

Figure 1.3 – Characters from the ASCII Set.

which came first; that is, if a name entered as 'John' and one entered as 'john' were both in a collection, which should come first if the names were sorted? In addition, should all of the upper-case letters come before all of the lower-case ones; or should 'J' be right next to 'j'?

- Upper case letters should be easily converted into lower case and vice versa. In the standard encoding system used for 50 years or so, named ASCII, a lower-case letter is a numeric value of 32 larger than the corresponding uppercase value.
- The space character has to come before others to make sorting easy. It must be smaller than the 'A'.
- Non-alphanumeric characters were positioned to correspond to their shifted position on typewriters. Initially this made sense, but now typewriters are nearly obsolete and keyboards do vary some.
- The first two columns (32 positions) were reserved for control characters. Again, this is because when the system was devised, certain characters were used to indicate specific communications events. In addition to page ejections, tabs, and other carriage controls there were characters used to indicate such actions as the beginning of a header (SOH), beginning of text (STX), ringing the bell (BEL) and other things that were used in the days of teletypes.

- The digits 0–9, were placed so they correspond to values in binary prefixed with 011, meaning they are offset by 48 from their numerical value, making certain conversions straightforward. The digits also really need to be consecutive and in order.

On a computer, the letter 'A' is stored as the number 65, 'B' is 66, and so on. 65+32 = 97, which is the code for 'a'. The number '0' is 48, '1' is 49. This is one of a few distinct numerical codes for letters and other character, but to be used on a computer such codes must exist.

A string, or text, is stored as a set of adjacent numbers. For example:

```
84 104 105 115 32 105 115 32
97 32 115 116 114 105 110 103
```

is code for the string "This is a string." The quotes, of course, are not part of the string, so how does the computer tell where the string begins and ends? How are things stored on a computer? As numbers.

There are two common ways to represent a string. To have a sequence of characters preceded by the length of the string, as a number, and to have a special character that always means 'end of string' and has no other purpose. The string "This is a string" has 16 characters. The special character meaning 'end of string' is usually the character with code 0. So the two ways to represent the string are:

	16	
T	52	52
h	104	104
i	105	105
s	115	115
	32	32
i	105	105
s	115	115
	32	32
a	97	97
	32	32
s	115	115
t	116	116
r	114	114
I	105	105
n	110	110
g	103	103
		0

These are called representations of the string. All non-numeric data processed by a computer must have a numeric representation. Coming up with a numeric representation is a significant part of describing a complex scene or artwork.

Characters and strings are an essential means of communication between computers and human beings. The keyboard is specifically designed to send characters to the computer, and video screens used to be equipped only for displaying character. Now screens have been designed to display graphics, and displaying characters is done by *drawing* them. Characters each have an image that represents

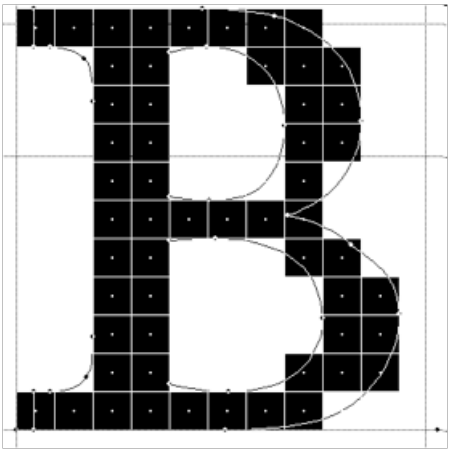


Figure 1.4 – Glyph for the letter 'B'

them. It's called a glyph. A glyph is a simple graphic. The letter 'B' could be is drawn as the above, Figure 1.4.

The paper is white, and the drawn glyph consists of black 'spots' drawn by the printer on a 2D mesh or grid. The black spots are called picture elements, or pixels for short. A computer screen consists of a great number of pixels in a grid, and each one can be indexed or accessed by specifying the row and column of the pixel that is being targeted.

Each character has a distinct shape, and when drawn as black ink on white paper can be used to create an image. Imagine that each character fits into a standard sized box on a page. A font of this type is called fixed-pitch or monospaced; the Courier font is an example. The box that encloses the character has black ink on a white background, and the amount of ink depends on the specific character. A space has no ink and is 'white'; a period ('.') has more black, but is mostly white, and the '@' character is mostly black with a little white. So, the character 'M' is 25.9% black. The character 'm' is 21.9% black. Characters can be placed in ascending order in terms of how much white or black is present and then used to draw a picture by drawing a dark character in a dark region of the image, and a white character in the lighter regions.

A critical element in programming is to find a representation for the data that you want to manipulate. There are two representations for a character described here: one is as a number that uniquely identifies what character is being discussed. The other is a visual representation of that character, one that can be drawn on a computer screen and recognized by a human. Others are possible: auditory representations (speak the name of the letter B), tactile (Braille), and so on.

Computer Programs

Computers can only do a very few things and can only operate on numbers. The key part of a representation is the way that the data (image, sound, drawing) is represented as numbers.

A computer program is a sequence of statements in a special language (a computer programming language) that describe how to accomplish a task. These statements are translated into computer instructions that do the same thing and that can be executed directly on the computer device.

Figure 1.5 – Mona Lisa drawn with characters on paper

Computer languages are based on human languages, in that they use human idioms that can be used to describe how to modify and examine data. For example, computer programming languages usually have a statement that says: **IF something is true THEN do something**

This is clearly based on a human language sentence. Something like *"if the light is red then stop"* would be understood by everyone. When using a computer, it works in very much the same way. Let's learn a programming language that can be used to do visual art. That language is called **Processing**. It was originally designed to be used to create visual artworks, but does have some of the same complexities and difficulties of other programming languages. Still, unless you can write a program you can't be a fully fledged generative artist. Let's push ahead.

Basic Principles of Generative Art

Drawing, painting, and sculpture have known design and application principles. Generative art is relatively new. Opinions vary on its value and methodology. For the purposes of this specific discussion it would be useful to suggest some principles that characterize the theory and conduct of generative art. These will be in force for the discussion that follows.

1. Generative art is the **art of the algorithm**. The artist must define how the work is constructed, in careful and very specific terms. The description can include a random component, but one that has been designed into the algorithm so as to enhance some aspect of the work, rather than to have the machine 'create' something. In other words, the randomness must be very specifically applied to some aspects of the creation of the work.
2. The machine (computer) is relegated to the role of renderer, not creator. The artwork is defined well enough by the artist that their view is the one being visualized.
3. The computer is used as a tool, but in such a way that there is a programmable element. This precludes drawing made using Paint, Illustrator, or Photoshop software apps from being classed as generative.
4. Generative art can be interactive (responsive to the viewer) and/or dynamic (changing with time) but does not have to be.

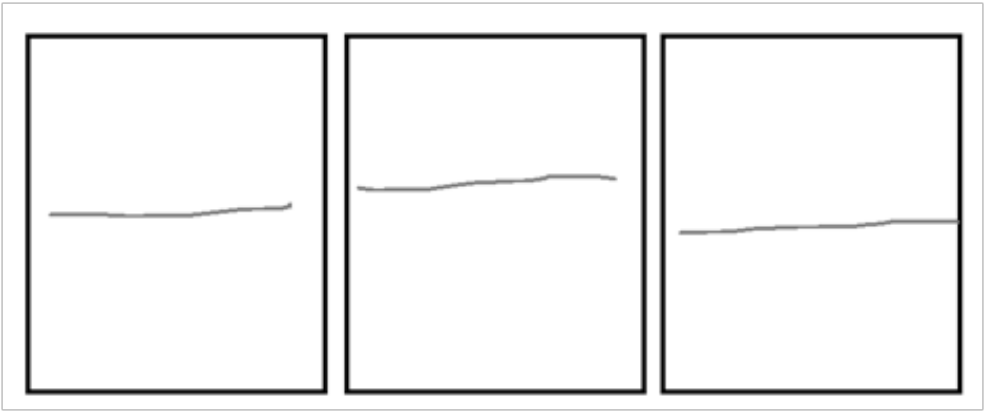


Figure 1.6 – Three Ways to render, "Draw a horizontal line across the center of the canvas."

5. Randomness in a generative work is under the control of the artist. The ultimate work is not completely random but can use randomness (as is the case in reality) to make variations in the rendering. No two drawings by the same artist are the same, but they have similar characteristics.
6. Two artists may have the same idea in mind, but their algorithms may differ in places so that the results differ too. Both may be satisfied with the result, even though the visual result differs.

Generative art is about describing what the work should look like, but unlike paintings and drawings it has to be extremely specific. Computers require a huge degree of precision, and that is a good thing and a bad thing. Artists don't work that way. "Draw, by hand, a horizontal line across the center of the canvas," one could say. Figure 1.6 are results that correspond to that description.

These drawings show quite a bit of variation. Computers, on the other hand, require precision. The same description when given to a computer could be "draw a line from the point (0, 300) to the point (500, 300)." The result would be perfectly straight and would be performed in the same way each time it was requested. This is one of the problems that generative art has, actually: the perfection of the rendering makes the result obviously artificial. Efforts to have generative artworks look as if humans have made them, should that be desirable, involve introducing various irregularities into the work. The description "Draw a horizontal line across the center of the canvas" would be converted into a collection of drawing operations that would give a slightly irregular line that was not quite horizontal or exactly at the center of the canvas, and that would be drawn a little differently

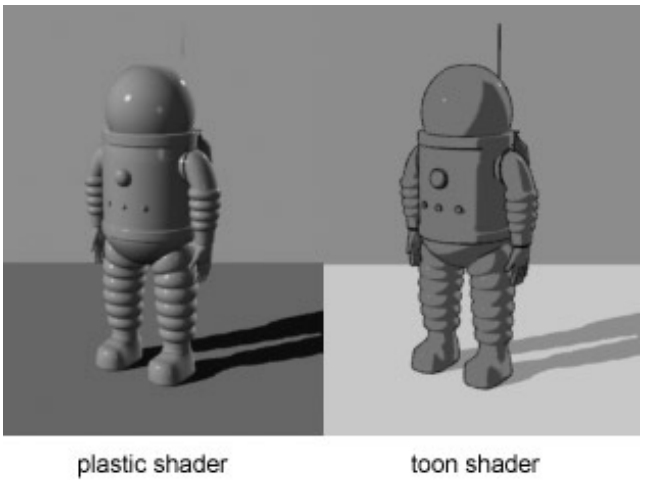


Figure 1.7 – Non-photorealistic rendering (Right) using a shading method called 'cel shading'.

each time. A degree of variation would likely appear in the color and thickness of the line as well.

Computer graphics are capable of producing images that are indistinguishable from a photograph. This is valuable when creating a motion picture or making high resolution images for advertising, but we all know that photography did not eliminate other forms of art when it was invented. People continued to draw, paint, and sculpt. Art is about conveying ideas and feelings, and is a fundamental human activity. People enjoy art that has been created by other people. Making a computer-rendered work appear as it would have if a human had rendered it is called non-photorealistic rendering, and it is a popular sub-field of computer graphics. It is not a required technique in generative art but is one of many that can be considered for any piece being made. It is a design feature.

Design

Artists sometimes spend a great deal of time working out how their painting or drawing will be laid out on the paper. They will draw sketches with the elements of the piece put in different places or seen from different perspectives. When they are happy with the result the final work can be started. This is a design phase, called a 'study' in the art community. Not all artworks and artists do this, but for generative art it is an absolute necessity.

The need for design arises because the work must be described in great detail to the computer, first through use of an *algorithm* and then through

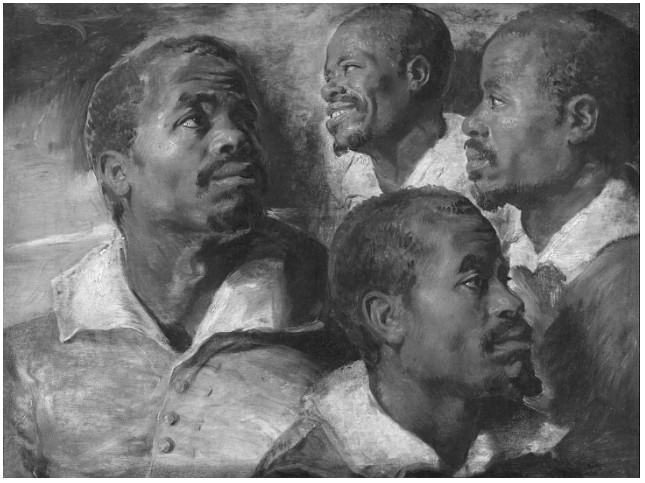


Figure 1.8 – An example of a 'study' in art. (Rubens, *Four Studies of a Head of a Moor*). For generative art, 'studies' are a necessity.

a computer program. The program is an embodiment or implementation of the artist's algorithm in a way that can be executed by the computer. The algorithm itself is a step-by-step set of directions for how to do something, in this case for how the resulting generative work will look.

Defining an algorithm and writing a program begin with a problem to be solved, and in generative art the problem is stated in visual terms, as clearly as possible. Examples may be:

• Draw a set of lines emanating from a single point that are 2 degrees apart and where each line is a small amount longer than the previous one. (Fig 1.9)

This description is not unambiguous. The number of lines to be drawn is unspecified, as is the initial line length. Is the angle changing in a clockwise or counter-clockwise direction? What are the foreground (line) and background colors? What are the coordinates of the lines' common point?

• Draw a set of circles of increasing radius where the center points differ by 6 units in a horizontal direction (Figure 1.10).

Again, there are many ways this could be drawn from this description. What is the initial center and radius, how much does the radius increase each time?

• Draw a red square and then a yellow circle so that the two shapes share a center point and the circle touches the square in four places (Figure 1.11).

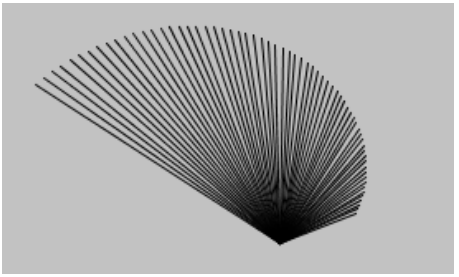


Figure 1.9

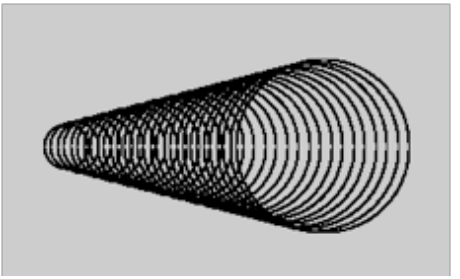


Figure 1.10

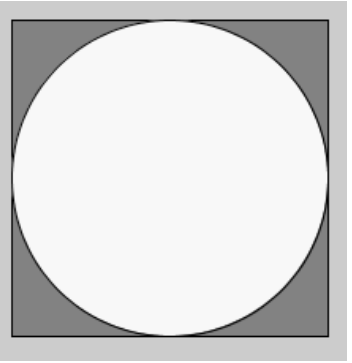


Figure 1.11

Note: See image in color pages of this book.

How big should the square/circle be? Where located in the drawing area?

When writing a program to draw something, the image needs to be described in great detail. This is not true of drawing and painting using other media. Computers describe everything with numbers and do things very precisely as a result. Artists generally do not. There is thus a difference between how artists are used to working and how they must work using a computer.

On the other hand, when using a computer the consequences of making an error are small, and the result can easily be changed. Consider the first example above in Figure 1.9. The initial line length for the figure was 50 and the initial angle was -21 degrees. In a single statement in the program we can change both properties and create a new picture (Figure 1.12).

This ability to edit an artwork is something that should be envied by painters and sculptors. A problem created by this is that there is really no *original* piece – all generative works can be duplicated precisely as many times as desired.

Communication: Essential to Generative Art

When someone programs a computer, they are really communicating with it. It is a very imperative and precise communication to be sure. *Imperative* because the computer has no choice; it is being told what to do and will do exactly that. *Precise* because a computer does not apply any interpretation to what it is being told. Computers do

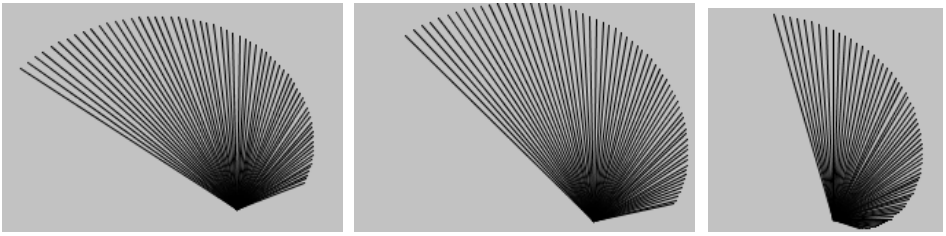


Figure 1.12 – Variations on the theme.

```
length = 50, angle = -21        length = 60, angle = -11      length = 20, angle = 20
```

not think, and so they can't evaluate a command that would amount to 'expose the patient to a fatal dose of radiation' with any skepticism. As a result, we, as programmers, must be careful and precise in what we instruct the machine to do.

When humans communicate with each other, we use a language. Similarly, humans use languages to communicate with computers. Such languages are artificial (humans invented them for this purpose, all at once), terse (there are few if any modifiers, no way to express emotions or graduations of any feeling), precise (each item in the language means one thing) and written (we do not *speak* to the computer in a programming language. Not yet; perhaps never).

The process begins with a problem to be solved, and the first step is to state the problem as clearly as possible and then analyze the problem and determine methods by which it may be solved. Computers can only directly manipulate numbers, and so it is common for solutions discussed at this stage to be numerical or mathematical. A sketch of the solution, perhaps on paper in a human language and math, is created. This is translated into computer language and then typed into computer form using a keyboard. The resulting text file is called a '*script*', '*source code*', or, more commonly, the '*computer program*'. A program called a **compiler** takes this program and converts it into a form that can be executed on the computer. Basically, all programs are converted into a set of numbers called machine code which the computer can execute.

In generative art, an artist does not touch a pen or brush to paper. The artist tells the computer how to create the artwork that they are seeing in their head. The artist also does not use a drawing tool like Paint or Photoshop but instructs the computer through a program. When using a pencil, an artist can sit in front of a blank paper and simply start to draw, not having a detailed plan but simply sketching what they see or moving in some desired visual direction. When using

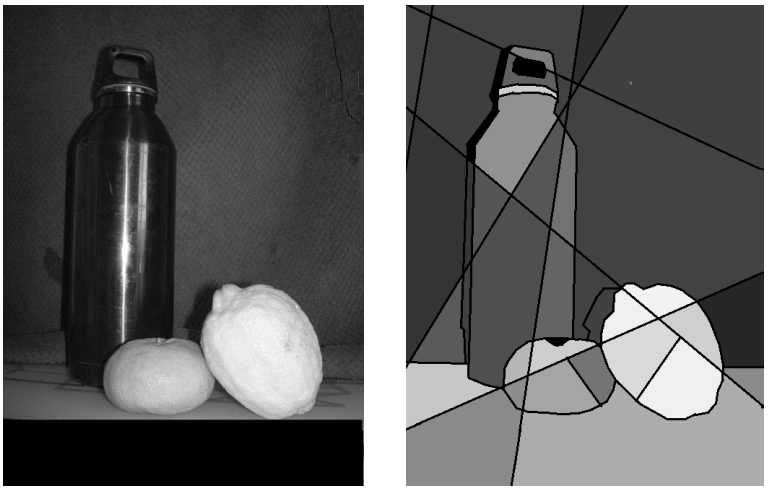


Figure 1.13 – A cubist-style rendering.

a computer, this is not productive. When someone begins to write code they must already have some outcome in mind. Creating art in this way is more like writing a story than painting a picture. A story has a detailed outline, or at least some direction and a way to get there. So does a computer program. The story is what we called the algorithm. To some degree, stories are all algorithms. *Hamlet* is written in a way that leads to a great many theatrical productions, each different. *Hamlet* is an algorithm, and a specific production is like a program.

A generative artist must usually have an algorithm in mind for creating their work, and this begins with a basic idea, as does most art. Where those ideas originate is a matter for philosophical discussion, but in generative art they generally take one of six forms:

1. "I have a detailed image in my mind."

In this case the artist knows what they want to create. It could be something as simple as a still life or landscape, or it could be abstract. But they have a picture in their head that they wish to put on paper (or the screen). If the idea is to represent the object precisely then, of course, a photograph would be better than other representations, so there is usually some particular interpretation that is intended for the rendering.

The artist must therefore devise a method (algorithm) for rendering the scene the way it is imagined.

The cubist-style piece in Figure 1.13 was done by first sketching the scene and then dividing the sketch into polygonal regions. Each polygon was thin-filled with a color similar to that seen in the scene.

Figure 1.14 – An impressionist rendering of a flower from an image.

2. "I have an actual image and wish to change it or use it."

This process is more common than the previous one. Working with actual images (photographs) involves a standard set of techniques. Think of these as transformations, in which image elements are modified systematically to produce new visual effects. An example would be to take an image of, say, a sunflower and try to make it look as if it has been painted by an impressionist; or start with a photograph of a person and transform it into a pencil sketch.

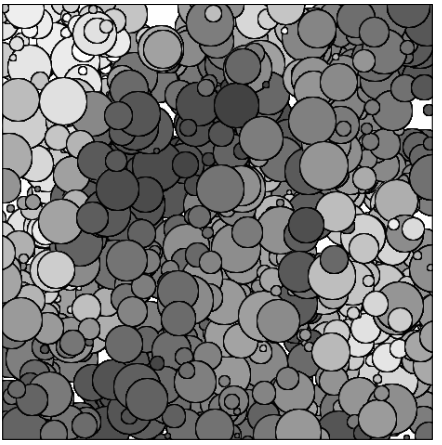
Algorithms at work here are those of image processing and analysis combined with some computer graphics techniques.

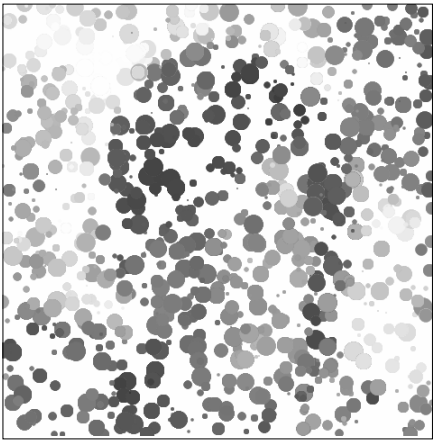
3. "I wonder what would happen if …"

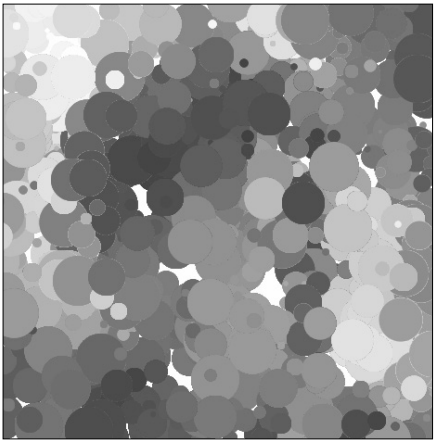
This is the domain of educated speculation. Artists with a good imagination and a grasp of algorithms and representation can visualize geometries and colors that would be difficult to paint or draw, but that have an instantiation in a digital space. Sometimes an idea works, sometimes it does not.

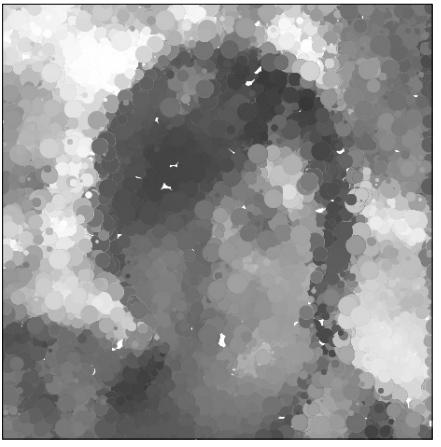
This is where generative art shines. A computer permits someone (an artist) to make mistakes, fix them, and try again many times an hour. The first idea may not be a winner but being able to see what it looks like and then being able to change it in a few minutes and try again makes it possible to force an idea into success.

Consider: what would an image be like if each point on an image was replaced with a circle that was fill with a color that was the average of the colors within that circle? Sounds interesting. The first try looks like Figure 1.15









(Top images) Figure 1.15 – Replacing pixels with colored circles.
(Center left) Figure 1.16 – (Left) Smaller circles, more white space. (Right) More circles.
Bottom: Figure 1.7– Even more circles.
Note: See images in color pages of this book.

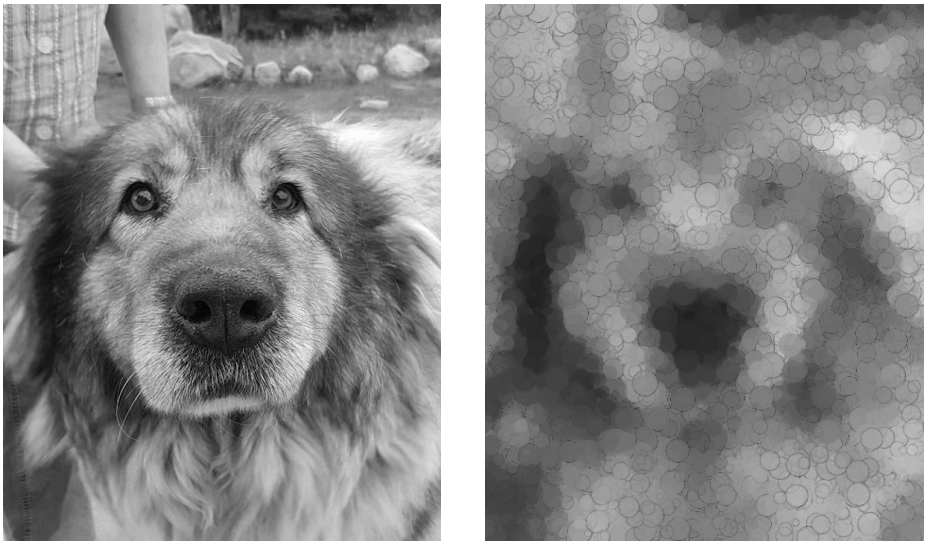


Figure 1.18 – A final transformation, as hoped for.

image on the right with image on the left being the original. There are problems. Perhaps the circles should not have a black outline. Fixing that results in experiments shown in 1.6 – 1.7. Better. Perhaps the circles are too large? Let's make them smaller – these are between 5 and 55 units in radius. Let's try 5 to 22: Perhaps when the circles get smaller we should have more of them. Right now there are 1000. Let's try 9000. There are a few white (unpainted) regions. Increasing the number of circles to 20000 will solve this problem in many cases but takes more time. That's usually okay, as this takes a few seconds to draw, and making it take 20 seconds it not a problem. If using 20000 circles does not work, simply run the program again. It may work this time. Things here have a random aspect.

The result is interesting but has the appearance of looking through cellophane. Maybe we should think about transparency. Overlapping circles could contribute to the color of the region. Perhaps this could apply to the circle outlines as well. Another thing about this program is that it can be applied to any image we like. So, Figure 1.18 is the new code applied to a new image.

If an artist wishes to create a new work, they start from scratch with their technique and spend hours or days making a new work. Major modifications are quite difficult, and if the work is not turning out as expected, one can start over or continue with the piece as is. Once a generative artist describes their methods as an algorithm it can be applied to any subject almost instantly. Modifications are nearly instantaneous. An operation like 'change the eye color' can be implemented in seconds. This is

Figure 1.19 – Frames from *Filament* – an animation of a static image. The lines look alive.

an amazing step forward. Artists can describe what they wish to do and then do that as many times as they like.

4. Make a static image into something dynamic.

This is a variation on #2, but in a new dimension not yet explored: time. A motion picture or animation consists of multiple still images (called *frames*) that are displayed in rapid succession. The concept is called *persistence of vision* and refers to the fact that an image stays in our perception for a brief moment after it is no longer there.

What does this have to do with generative art? Well, a good deal of generative art is dynamic, animated, or reactive. In particular, we can take a single still image and make it more dynamic by adding things to it or manipulating elements of the image with time. An example of this type of image manipulation can be seen in a work called *Filament,* on the book's *minkhollow.ca/gallery* web site. A still image of the lit filament of an electric lamp was modified to create many almost-imperceptibly different animation frames. It has the appearance of flickering and emitting small particles (Fig. 1.19).

There are many ways to impart motion to a still image: parts of the image can be made to move; parts can morph into new shapes; colors can change. And so on.

5. Allow the viewer to contribute

There are many ways for a viewer to contribute to a work, some more intentional than others. Obviously, buttons and sliders, mouse motions and keyboard inputs are standard inputs to a computer and offer the user very intentional and obvious choices. Drawing a line with the mouse, clicking on an object, controlling motion with a key press – these are standard interface methods these days.

What about using the motion of the user to direct a visual or auditory activity? Using the speed or position of bodies or parts, the number of people in the room or the colors of their clothing are only some ways that the audience can become essentially a part of the work. The originality and

Figure 1.20 – The number of people in the image is reflected in the artwork (fireworks).

effectiveness of the ideas an artist uses for this means the difference between something that is ordinary and something that is exciting.

Consider a piece that draws simulated fireworks. It looks at the audience using a camera, identifies how many people appear to be in the scene, and uses that number to create the same number of fireworks in the display. It may be a while before people notice what is happening, but they will eventually.

This is a form of audience interaction. In this case it is somewhat implicit, and the decision that will affect the piece was made long before the audience showed up; it was made when they got dressed that morning, in fact.

6. Merge different forms, such as sound and pictures

Sound is like animation, in the sense that it is time dependent. To be glib, if there were no such thing as time then all of the musical notes would happen at once. An image can be the inspiration for sound, it can be an interface for sound creation, or it can simply be a vehicle. Consider the example in Figure 1.20. People are wearing colored clothing, and those colors dictate the colors used in a graphic. Why not have them also dictate sounds as well? Certain color ranges could, for example, correspond to certain musical notes. Each note is the foundation for a scale, and notes on a particular scale are known to 'go together', as it were.

It would be possible to select colors to be notes on a scale. Or shapes could more to the tempo of music, or their motion could set the tempo. A very common and under-appreciated use of image and sound is in video games, where an object that exists only as a drawing appears to create a sound: a gun shot, or a collision for example.

This kind of generative art is difficult to describe effectively in a written book, but examples are posted on the *Generative Art* website at https://www.minkhollow.ca/parker/generative. ■

References, Chapter One

1 Cavell-Clarke, Tom Welch (2018). *How Computers Work*, Kidhaven.

2 Andrew Graham Dixon (2018). *Art: The Definitive Visual Guide*, New Edition. DK; 2 edition. ISBN-13: 978-1465474759.

3 Roger Ebert (2010). *Roger Ebert's Journal*. http://www.rogerebert.com/rogers-journal/video-games-can-never-be-art

4 Anne Gantefuhrer-Trier (2015). *Cubism*. TASCHEN. ISBN-13: 978-3836505390.

5 Nigel Lesmoir-Gordon, Will Rood, Ralph Edney (2009). *Introducing Fractals: A Graphic Guide*. Icon Books Ltd; Revised ed. edition (Oct. 2, 2009).

6 Ronald Lind, Dr. Joy Lind (2018) *How Computers Work: Computing Foundations*, Independently published. ISBN-13: 978-1980333173.

7 Mandelbrot, Benoit B. (1983).*The fractal geometry of nature*. Macmillan. ISBN 978-0-7167-1186-5.

8 Matt Pearson. (2011) *Generative Art: A Practical Guide Using Processing*. Manning Publications; 1 edition.

Fig. 10.3 – *Ellipsis, JR Parker.* Named for what it looks like (a sequence of dots), *Ellipsis* is not about dots or text. It is about motion, perspective, abstraction, and perception.

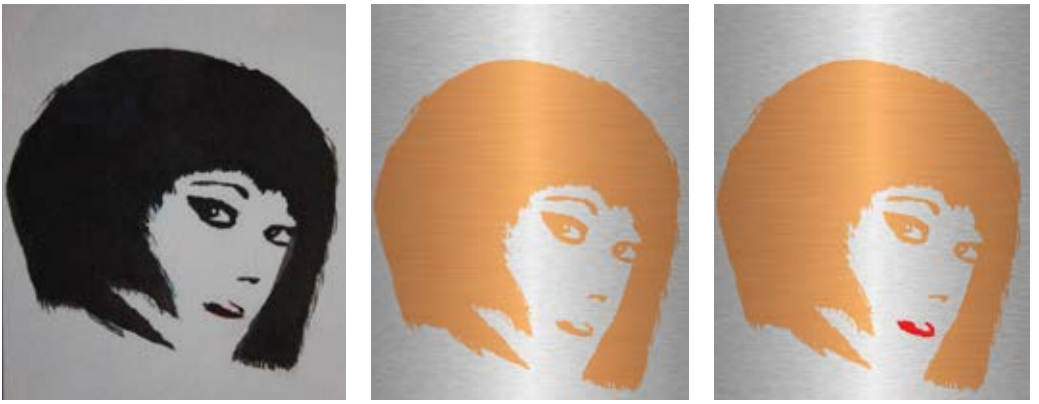


Fig. 10.10 Color – Metalized versions of the *Quorra, JR Parker.*

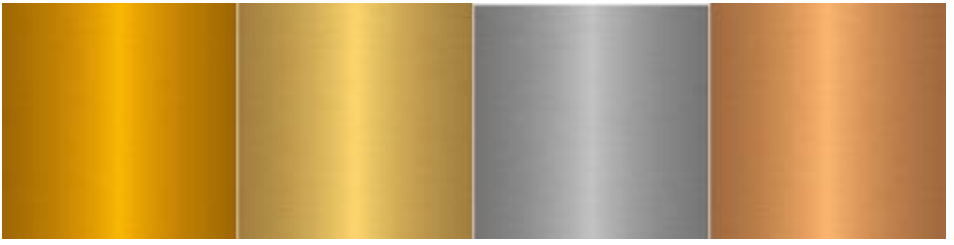


Fig. 10.8 Color – (Above) *Klaus, JR Parker.*
Fig. 10.9 Color – (Below) Metal texture images with changes to the hue.

Fig. 10.11 Color – Frame from source video.

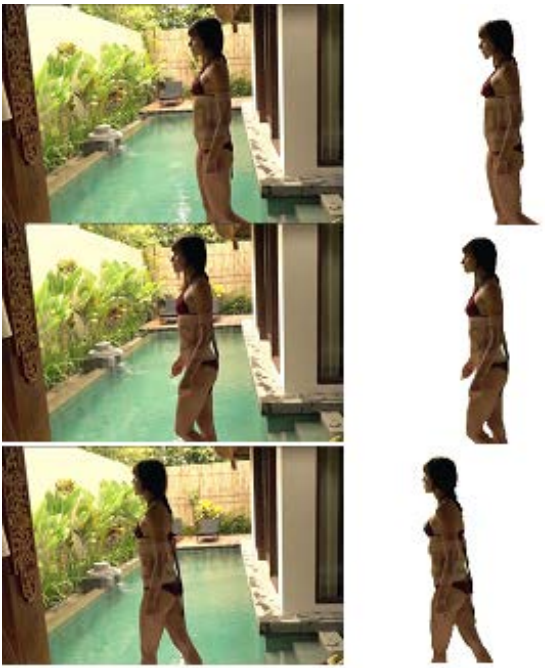
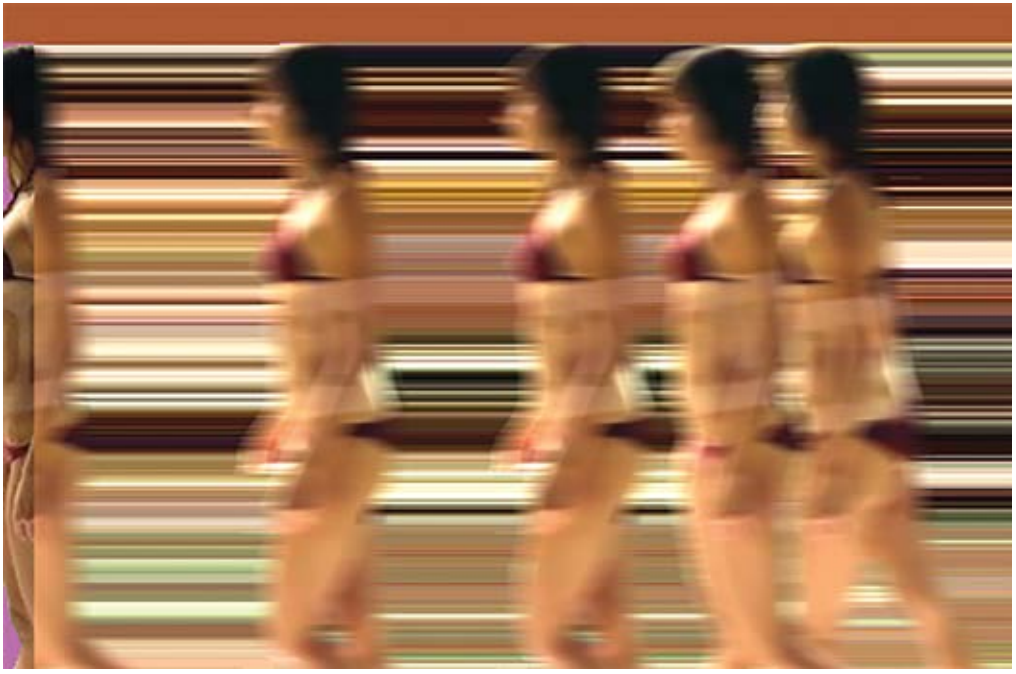


Fig. 10.12 Color – Three sample frames
from the video with woman's image extracted manually.

Fig. 10.13 Color – Next step, color is blurred between instances of woman's image.



Figures 10.14 Color – The final version of *Woman in the Burgundy Bikini, JR Parker.*

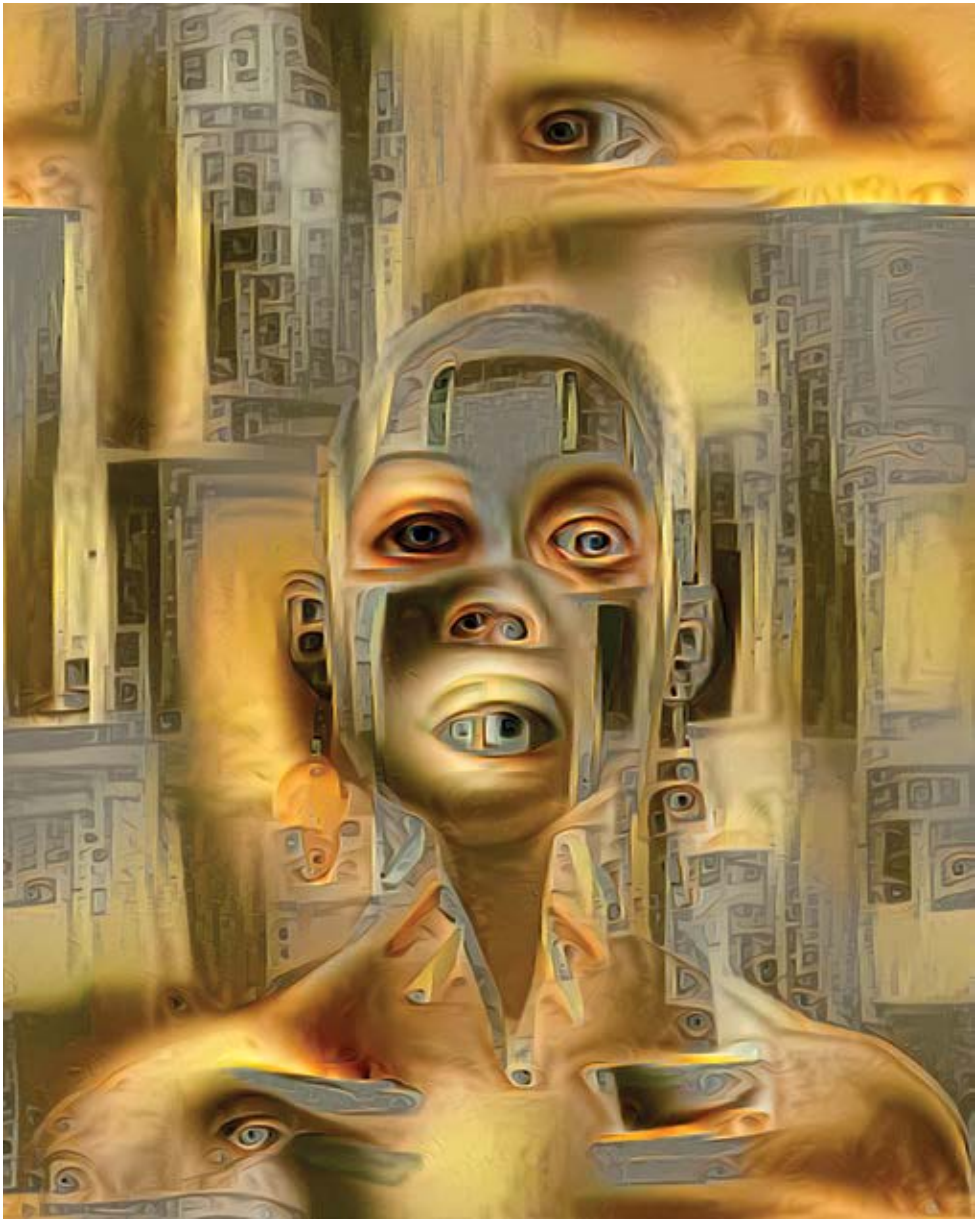
"Code Zebra" Site opening screen with "Leap" displayed.
CodeZebra.net is a chat visualization and dialogue tool created by Sara Diamond, 2003.

Detail of "Code Zebra" user interface 'post-Leap' with 'What Is It', 'Experience', 'Softwear', 'Software', and 'Games' navigation buttons disclosed. (www.CodeZebra.net)

"Terminal Velocity" Steve DiPaola, 2019.

"Totem" Steve DiPaola, 2019.

"Untitled" Steve DiPaola, 2019.



"The City Emerges" Steve DiPaola and Graeme McCaig, 2019.

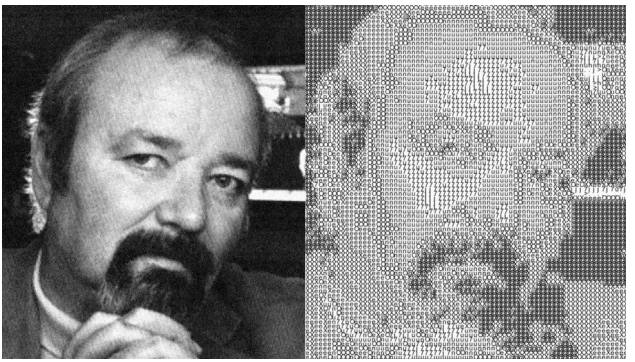
Generative Art: Algorithms as Artistic Tool
by James R. Parker; Foreword by Sara Diamond

UPROUTE Imprint of Durvile Publications
ART046000 Art/Digital
6.25 in. x 9 in. | 272 pages
ISBN: 978-1-988824-38-3
Price $37.50 Can.; $29.95 US | Color & b/w pics
Book release date: January 1, 2020
Individuals : Ask your bookseller or buy online.
Booksellers: Order through University of Toronto (UTP)

Book Websites

https://www.durvile.com
https: www.genart.ca

Durvile & UpRoute Books
Durvile Publications Ltd.
Calgary, Alberta, Canada
E-mail: info@durvile.com, Ph: 403 818-4808



About the Author, James R. Parker PhD

Dr. J.R. Parker studied mathematics and computer science before ending up as a Professor of Art at the University of Calgary. His expertise ranges from computer simulation, image processing, artificial intelligence, game design, and generative art. When asked, he claims to be an expert in 'complex interacting systems', but don't ask him to tie a half hitch. He lives at Mink Hollow, a small ranch in the foothills of the Rocky Mountains where he helps raise small animals and Tennessee Walking Horses.

About the Cover Artist, Steve DiPaola PhD

Using Cognitive Science as a basis for artistic work, Dr. Steve DiPaola, and his IVIZLAB research team at Simon Fraser University, attempt to model aspects of human creativity in AI. Specially they use Neural Networks (and evolutionary systems) in the form of Deep Learning, CNNs, RNNs and other modern techniques of Generative Design to model aspects of human expression and creativity.

"In an era in which Artificial Intelligence is positioned as both existential threat and transformational opportunity this book uses computational tools to augment and express human imagination, without supplanting human expression and experimentation."

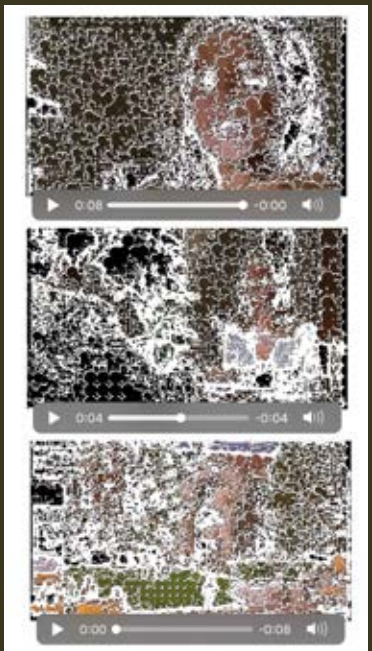
— SARA DIAMOND PHD

President, OCAD University, Toronto. From the Foreword.

Generative art is the art of the algorithm where artists must carefully design the nature of their work, and then implement it as a computer program. In the book, J.R. Parker presents computer programming concepts and generative art principles as a way to create algorithmic computer art using art and design best practices. In addition, readers have access to program codes and video tutorials through the book's web site.

JAMES R. PARKER

is Professor of Art *at the* University of Calgary *whose* expertise ranges *from* computer simulation, image processing, artificial intelligence, game design, & generative art.



Frames from *Geome*, J.R. Parker.
Transforming images using natural color patterns.

KEYWORDS

Digital Art, Algorithms, Generative Art, Artificial Intelligence, Computer Interface, Image Processing, Visual Coding, Animation, Web Art

Cover Art
"The City Ponders"
Steve DiPaola PhD
& Graeme McCaig

Cover & Page Design
Lorene Shyba PhD

DURVILE & UpRoute Books

DURVILE.COM

ART046000 Art/Digital

$29.95 US | $37.50 Canada

9 781988 824383

MINKHOLLOW.CA/PARKER/GENERATIVE