Processing

Creative Coding and Computational Art

Ira Greenberg

friendsof
DESIGNER TO DESIGNER™
an Apress® company

Processing: Creative Coding and Computational Art

ISBN-13: 978-1-59059-617-3

ISBN-10: 1-59059-617-X

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit www.apress.com.

The source code for this book is freely available to readers at www.friendsofed.com in the Downloads section.

Credits

Lead Editor
Chris Mills

Technical Editor
Charles E. Brown

Technical Reviewers
Carole Katz, Mark Napier

Editorial Board
Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick, Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Jeff Pepper, Dominic Shakeshaft, Matt Wade

Project Manager
Sofia Marchant

Copy Edit Manager
Nicole Flores

Copy Editor
Damon Larson

Assistant Production Director
Kari Brooks-Copony

Production Editor
Ellie Fountain

Compositor
Dina Quan

Artist
Milne Design Services, LLC

Proofreaders
Linda Seifert and Nancy Sixsmith

Indexer
John Collin

Interior and Cover Designer
Kurt Krames

Manufacturing Director
Tom Debolski

To Robin, Ian, and Sophie.

CONTENTS AT A GLANCE

CONTENTS

FOREWORD

If you are like me (and the fact that you are holding a Processing book in your hands indicates there's a fair chance that you are), then a quick flip through the pages of this book, glancing at the many illustrations, should be enough to set your heart beating just a little bit faster, and start seeds of ideas sprouting in your head.

Processing is a richly visual language, which is pretty obvious if you've performed the aforementioned page flipping. It has its roots in a language called Design by Numbers, developed by Professor John Maeda at MIT, and was in fact created by two of Maeda's students, Ben Fry and Casey Reas. Whereas most languages are built to create serious applications, Processing almost seems to have been created to just have fun with. The language has been used to create various data visualization and installation art pieces, but most often you just see people playing with it, creating complex and beautiful pictures and animations. As a matter of fact, you don't even make Processing applications; you make sketches—which go in your sketchbook. This aspect of the language has drawn many creative coders who blur the boundaries between programming and art.

Many like to draw a comparison between Processing and Adobe (née Macromedia) Flash, a commercial program often used to create graphically rich, often purely experimental animations using ActionScript, an easy-to-learn programming language. Indeed, many of the people using Processing started out programming in Flash, and switched to take advantage of the superior speed and performance, additional commands, and flexibility of Processing. Although Flash has gained a lot over the years in terms of performance and capabilities, Processing remains the tool of choice for many artist-coders.

Processing has grown quite a bit over the years. It's an evolving language, added onto by various plug-ins and contributions from a dedicated community. It's deceivingly simple, allowing you to get started quickly, but it provides an incredible amount of depth for those who care to peek beneath the surface.

Although there are various online resources, Processing has lacked a printed book of any sort. This book fills that gap, and then some. In the tradition of the language, this book covers both the artistic and the programming aspects of Processing. And if you are stronger on the art side than the code side, fear not. The author leads you into it gently, giving you just

the bits you need to get started. On the other hand, when you are ready to dive in deep, there's more than enough material to keep you up late at night coding.

So take another flip through the book for inspiration, take a deep breath, get comfortable, and dive in, just like I'll be doing as soon as I finish writing this!

Keith Peters, April 2007

ABOUT THE AUTHOR

Photo by Robin McLennan

With an eclectic background combining elements of painting and programming, **Ira Greenberg** has been a painter, 2D and 3D animator, print designer, web and interactive designer/developer, programmer, art director, creative director, managing director, art professor, and now author. He holds a BFA from Cornell University and an MFA from the University of Pennsylvania.

Ira has steadily exhibited his work, consulted within industry, and lectured widely throughout his career. He was affiliated with the Flywheel Gallery in Piermont, New York, and the Bowery Gallery in New York City. He was a managing director and creative director for H2O Associates in New York's Silicon Alley, where he helped build a new media division during the golden days of the dot-com boom and then bust—barely parachuting back to safety in the ivory tower. Since then, he has been inciting students to create inspirational new media art; lecturing; and holding residencies at numerous institutions, including Seton Hall University; Monmouth University; University of California, Santa Barbara; Kutztown University; Moravian College; Northampton Community College's Digital Art Institute; Lafayette College; Lehigh University; the Art Institute of Seattle; Studio Art Centers International (in Florence, Italy); and the City and Guilds of London Art School.

Currently, Ira is Associate Professor at Miami University (Ohio), where he has a joint appointment within the School of Fine Arts and Interactive Media Studies program. He is also an affiliate member of the Department of Computer Science and Systems Analysis. His research interests include aesthetics and computation, expressive programming, emergent forms, net-based art, artificial intelligence, physical computing, and computer art pedagogy (and anything else that tickles his fancy). During the last few years, he has been torturing defenseless art students with trigonometry, algorithms, and object-oriented programming, and is excited to spread this passion to the rest of the world.

Ira lives in charming Oxford, Ohio with his wife, Robin; his son, Ian; his daughter, Sophie; their squirrel-obsessed dog, Heidi; and their night prowler cat, Moonshadow.

ABOUT THE TECH REVIEWERS

Carole Katz holds an AB in English and American Literature from Brown University. Her career as a graphic designer and technical communicator has spanned more than 20 years, including stints at small nonprofits, design firms, government agencies, and multinational corporations. Beginning with PageMaker 1 and MacDraw in the mid-1980s, Carole has used many types of software in a variety of design disciplines, including corporate identity, technical illustration, book design, and cartography. She is currently a freelance graphic designer, and lives with her family in Oxford, Ohio.

Mark Napier, painter turned digital artist, is one of the early pioneers of Internet art. Through such works as *The Shredder*, *Digital Landfill*, and *Feed*, he explores the potential of a new medium in a worldwide public space and as an engaging interactive experience. Drawing on his experience as a software developer, Napier explores the software interface as an expressive form, and invites the visitor to participate in the work. His online studio, `www.potatoland.org`, is an open playground of interactive artwork. Napier has created a wide range of projects that appropriate the data of the Web, transforming content into abstraction, text into graphics, and information into art. His works have been included in many leading exhibitions of digital art, including the Whitney Museum of American Art Biennial Exhibition, the Whitney's Data Dynamics exhibition, the San Francisco Museum of Modern Art's (SFMOMA) 010101: Art in Technological Times, and ZKM's (Center for Art and Media in Karlsruhe, Germany) net_condition exhibition. He has been a recipient of grants from Creative Capital, NYFA, and the Greenwall Foundation, and has been commissioned to create artwork by SFMOMA, the Whitney Museum, and the Guggenheim.

ACKNOWLEDGMENTS

I am very fortunate to know and work with so many kind, smart, and generous people. Here are just a few who have helped make this book possible:

Advisors, colleagues, and reviewers: Fred Green, Andres Wanner, Paul Fishwick, Paul Catanese, Mary Flanagan, Laura Mandell, Scott Crass, Mike Zmuda, and David Wicks for showing an interest when it really, really mattered; technical reviewers Carole Katz, Mark Napier, and Charles E. Brown for helping me simplify, clarify, and rectify—the book is far better because of your combined wisdom; my wonderful colleagues and students at Miami University, in the Department of Art and Interactive Media Studies program—especially Mike McCollum, Jim Coyle, Bettina Fabos, Glenn Platt, Peg Faimon, and dele jegede—for tolerating such a looooong journey and my perpetual "when the book is finished" response.

The wonderful people at friends of ED: Production editor Ellie Fountain for always responding kindly to my neurotic, 11th-hour requests; copy editor Damon Larson for his patience and precision in helping me craft actual grammatical sentences; project manager Sofia Marchant for keeping the book (and me) from slipping into the procrastinator's abyss—I couldn't have pulled this off without you! Lead editor and heavy metal warrior Chris Mills for believing in a first-time author and providing constant support and sage advice throughout the entire process. I appreciate this opportunity more than you know, Chris!

The wonderful Processing community—especially Ben Fry and Casey Reas for giving me something to actually write about. I know I am just one of many who owe you a world of thanks for selflessly creating this amazing tool/medium/environment/language/revolution.

My incredible mentors, friends, and family: Petra T. D. Chu, for all your generosity and support over the years; Tom Shillea, Bruce Wall, and Sherman Finch for helping plant the "creative coding" seed; Bill Hudders for sticking around even after I put down the paintbrush; Roger Braimon for keeping me from taking anything too seriously; Jim and Nancy for moving 700 miles to join us in a cornfield; Paula and Stu for giving me (and my Quadra 950) our first shot; my uncles Ron and Ed and their respective wonderful families for fostering my early interest in science and technology and the belief that I could do it "my way"; Bill and Rae Ann, for lovingly supporting the west coast surf and burrito operations; Ellen, Sarah, Danny, Ethan, Jack, Anne, Miles, Shelley, Connor, and Matthew for all your kindness and love over so many years; my genius brother Eric, for keeping me humble and bailing me out on

(way) more than one occasion—you're a real hero; my parents for tolerating (and even supporting) years of artistic indulgence and always, always being there for me; my delightfully mischievous and beautiful children, Ian and Sophie, for letting daddy stare at his laptop all day and night, while having their own screen time severely limited; and most importantly my brilliant and infinitely kind wife, Robin, for being a constant source of encouragement and peaceful joy in my life. I love you bel!

INTRODUCTION

Welcome to *Processing: Creative Coding and Computational Art*. You're well on your way to becoming a Processing guru! All right, maybe it will take a bit more reading, but with Processing, you'll be cranking out creative code sooner than you think. Best of all, you'll be creating as you learn. Processing is the first full-featured programming language and environment to be created by artists for artists. It grew out of the legendary MIT Media Lab, led by two grad students, Casey Reas and Ben Fry, who wanted to find a better way to write code that supported and even inspired the creative process. They also wanted to develop an accessible, affordable, and powerful open source tool; so they decided to make the software available for *free*.

Casey and Ben began developing Processing in the fall of 2001, releasing early alpha versions of the software soon after. In April 2005, they released the beta version for Processing 1.0. To date, over 125,000 people have had downloaded the Processing software, and Ben and Casey had been awarded a Prix Ars Electronica Golden Nica, the electronic/cyber-arts version of an Oscar. In addition, many leading universities around the world have begun including Processing in their digital arts curriculum, including Parsons School of Design; Bandung Institute of Technology, Indonesia; UCLA; Yale; NYU; Helsinki University; Royal Danish Academy of Fine Arts, Copenhagen; School of the Art Institute of Chicago; Miami University of Ohio; University of Washington; and Elisava School of Design, Barcelona (and many, many others).

Yet, in spite of all of Processing's phenomenal success, its story is really just beginning. As of this writing, version 1.0 of the software is on the brink of being released, as are the first few books on the subject. There are even people (as shocking as this sounds) who still haven't heard of Processing. So rest assured, it's still not too late to claim Processing pioneer status. Processing has a very bright future, and I'm excited to be able to introduce you to creative coding with this amazing language.

Impetus for writing the book

If you're anything like me (and I suspect you are since you're reading this book), you are a creatively driven individual—meaning that you do give a damn about how things look,

sound, feel, and so on, besides just how they function. I suspect you also learn best in a nontraditional way. Well, if this describes you at all, you've picked up the right book. If, on the other hand, you pride yourself on your robotic ability to tune out sensory data and follow linear directions, then (1) keep reading, (2) make art, and (3) buy multiple copies of this book.

My own interest in writing code evolved organically out of my work as a painter and designer over a long period (a well-timed, nonserious illness also contributed). I graduated with my MFA in painting in 1992, and got a teaching job right out of grad school. However, I soon realized that I wasn't ready to teach (or hold a job for that matter), and landed up quitting within a couple of months (my folks weren't too pleased at the time). Fortunately, an uncle of mine (the previous black sheep in the family) stepped in and suggested I look into computer graphics. With nothing to lose, I rented a Mac 2ci, borrowed some software, and locked myself away for a couple of months.

I eventually developed some basic skills in digital imaging, page layout, and vector-based drawing. I also began studying graphic design, which, despite my two overpriced degrees in painting, I knew next to nothing about. Equipped with my new (very shaky) skills, I put together some samples and went looking for work. Over the next few years, I got involved in a number of startups (most quickly imploded), as well as my own freelance design business. The work included print, CD-ROM/kiosks, 2D and 3D animation, video, broadcast, and eventually web design. Throughout this period, I also continued to paint and show my work, and began teaching again as well.

My paintings at the time were perceptually-based—which means I looked at stuff as I painted. I worked originally from the landscape, which eventually became just trees, and then a single tree, and finally branches and leaves. The paintings ultimately became purely abstract fields of color and marks. This transformation in the painting took a couple of years, and throughout this period I worked as a designer and multimedia developer. I was dealing with a fair amount of code in my multimedia work, but I still didn't really know how to program, although I had gotten really adept at hacking existing code. I suspect this may sound familiar to some readers.

Then I got ill and was laid up, which turned out to be the perfect opportunity to learn how to program. I'll never forget the first program I hacked out, based on the pattern structure in one of my field paintings. The program wasn't pretty, but I was able to translate the color field pattern in the painting to code and generate a screen-based approximation of the painting. But the really exciting thing (or disturbing thing, depending upon your perspective) happened when I was able to generate hundreds of painting variations by simply changing some of the values in the program. I remember excitedly showing what I had done to some of my more purist artist friends—who've since stopped calling.

It wasn't long before I was completely hooked on programming and was using it as a primary creative medium. I also began covering it more and more in my design courses, eventually developing a semester-long class on creative coding for artists. This book grows directly out of this experience of teaching programming to art students.

Intended audience

This book presents an introduction to programming using the Processing language and is intended as an entry-level programming book—no prior programming experience is required. I do assume, though, that you have some experience working with graphics application software (such as Adobe Photoshop) and of course some design, art, or visualization interests—which although not necessary to read the book, makes life more interesting. I *don't* expect you to "be good at" or even like math, but I'd like you to at least be open to the remote possibility that math doesn't have to suck—more on this shortly.

Coding as an organic, creative, and cathartic process

When I tell people I write code as my main artistic medium, they smile politely and quickly change the subject, or they tell me about their job-seeking cousin who makes videos using iMovie. For nonprogrammers, code is a mysterious and intimidating construct that gets grouped into the category of things too complicated, geeky, or time-consuming to be worth learning. At the other extreme, for some professional programmers, code is seen only as a tool to solve a technical problem—certainly not a creative medium.

There is another path—a path perhaps harder to maneuver, but ultimately more rewarding than either the path of avoidance or detachment—a holistic "middle" way. This is the path the book promotes; it presents the practice of coding as an art form/art practice, rather than simply a means to an end. Although there are times when a project is scoped out, and we are simply trying to implement it, most of the time as artists, we are trying to find our way in the process of creating a project. This approach of finding and searching is one of the things that makes the artist's journey distinctive and allows new unexpected solutions to be found. It is possible to do this in coding as well, and the Processing language facilitates and encourages such a "creative coding" approach.

"I'm an artist—I don't do math"

Early on in school we're put into little camps: the good spellers/readers, the mathletes, the artsy crowd, the jocks, and so on. These labels stick to us throughout our lives, most often limiting us rather than providing any positive guidance. Of course, the other *less* positive labels (poor speller, bad at math, tone deaf, etc.) also stick, and maybe with even more force. From a purely utilitarian standpoint, these labels are efficient, allowing administrators and computers to schedule and route us through the system. From a humanistic standpoint, these labels greatly reduce our true capabilities and complexity down to a few keywords. And worst of all, people start believing these limited views about themselves.

A favorite lecture I give to my art students is on trigonometry. Just saying the word *trigonometry* makes many of the art students squirm in their seats, thinking "is he

serious?" When I was an art student, I would have reacted the same way. And I remember studying trig in high school and not getting its relevance *at all*. Also, lacking discipline, I wasn't very capable of just taking my trig medicine like a good patient. So basically, I've had to teach myself trigonometry again. However, what I got this time was how absolutely fascinating and relevant trig (and math in general) is, especially for visually modeling organic motion and other natural phenomena—from the gentle rolling waves of the ocean, to a complex swarm, to the curvilinear structure of a seashell. Math really can be an expressive and creative medium (but perhaps not in high school). Finally, and likely most reassuring to some readers, playing with math in Processing is pretty darn easy—no proofs or cramming required.

Toward a left/right brain integration

I once had a teacher who said something to the effect that there is significance in the things that bore us, and ultimately these are the things that we should study. I thought at the time that he was being annoyingly pretentious. However, I've come to recognize something important in his suggestion. I don't necessarily think we need to study all the things that bore us. But I do think that at times, the feeling of boredom may be as much a defense mechanism as it is a real indicator of how we truly feel about something. I've become aware of the feeling of boredom in my own process, and notice it usually occurring when there is fear or anxiety about the work I'm doing (or the pressure I'm putting on myself). However, when I push through the boredom and get into a flow, I'm usually fine. I've heard many artists talk about the difficulty they have in getting started in their studios, spending too much time procrastinating. I think procrastination also relates to this notion of boredom as defense mechanism. My (unproven) hypothesis is that we sometimes feel boredom when we're stretching our brains, almost like a muscular reflex. The boredom is the brain's attempt to maintain the status quo. However, making art is never about the status quo.

Dealing with subjects like programming and math also seems to generate the sensation of boredom in people. Some people find it uncomfortable to think too intensely about analytical abstractions. I don't think this phenomenon has anything to do with one's innate intelligence; it just seems we each develop cognitive patterns that are hard to change, especially as we get older. As I've learned programming over the years, I've experienced a lot of these boredom sensations. At times, I've even (theatrically) wondered how far can I stretch my brain without going bonkers. I think it is especially scary for some of us to develop the less-dominant sides of our minds (or personalities). As artists, that is often (but certainly not always) the left side of our brain (the analytical side). However, I firmly believe that we will be more self-actualized if we can achieve a left/right brain integration. I even conjecture that the world would be a better place if more people perceived their reality through an integrated mind—so make code art and save the world!

Well, enough of my blathering. Let's start Processing!

Setting up Processing

If you haven't already downloaded Processing, you should do so now. You'll (obviously) need a working copy of the software/language to follow the tutorials throughout the book. To download the latest version, go to `http://processing.org/download/index.html`.

If you're not sure which version to download, keep reading.

> *As of this writing, the latest downloadable version of the software is 0124 BETA, released February 4, 2007. It's possible, by the time you're reading this, that the release number has changed, as the developers are in the process of stabilizing the current beta release as version 1.0. Any changes made to the language between beta release 0124 and version 1.0 should be very minor and primarily focused on debugging existing functionality. For more information about the different releases, check out* `http://processing.org/download/revisions.txt`.

Since Processing is a Java application, any platform that can run Java can theoretically run Processing. However, Processing is only officially released for Windows, Mac OS X, and Linux, and the software is only extensively tested on Windows and OS X. Linux users are somewhat on their own. Here's what the Processing site says in regard to Linux users:

> *For the Linux version, you guys can support yourselves. If you're enough of a hacker weenie to get a Linux box set up, you oughta know what's going on. For lack of time, we won't be testing extensively under Linux, but would be really happy to hear about any bugs or issues you might run into . . . so we can fix them.*

For more details about platform support, please check out `http://processing.org/reference/environment/platforms.html#supported`.

In selecting a version to download, Mac and Linux users have only one choice; Windows users have two choices: Processing with or without Java. The recommendation is to download Processing with Java. However, the without-Java version is available if download size is an issue and you know you have Java installed. If you're not sure whether you have Java installed, and/or the idea of changing your `PATH` variable gives you the willies, please download Processing with Java. If you still want to download them separately, here's a link (but remember, you've been warned): `http://java.sun.com/javase/downloads/index.jsp`.

OS X users already have Java installed, thanks to the good people at Apple.

Regarding Java, the most current version available on Windows is Java SE 6 (the SE stands for Standard Edition). On OS X, Java releases typically lag behind, and the most current version is J2SE 5 (the names are also annoyingly a little different). The most current version on Linux is also J2SE 5. If all this isn't confusing enough, Processing only supports J2SE 1.4

and earlier (yes, J2SE 5 and Java SE 6 come after J2SE 1.4). Version 1.4 is the version that comes bundled with Processing's Windows installer, and it is also the default installation in OS X. The reason Java versioning numbers go from 1.4 to 5 is because Sun, in their wisdom, decided to drop the "1." from the names—if you really care why, you can read about it here: http://java.sun.com/j2se/1.5.0/docs/relnotes/version-5.0.html.

What all this means to Processing users is that you can't use any new Java syntax specified in releases after 1.4 within the Processing development environment. (Syntax is essentially the grammar you use when you write code—which you'll learn all about in Chapter 3.) For the latest information on the tempestuous love affair between Processing and Java, please see http://processing.org/faq.html#java.

Web capability

Java's capabilities also extend to the browser environment, allowing Java programs (applets) to be run in Java-enabled browsers, similar to the way Flash programs run within the browser. Processing takes advantage of this capability, allowing Processing sketches that you create within the Processing development environment to be exported as standard Java applets that run within the browser.

One of the factors in Processing's quickly spreading popularity is its web presence. Processing's online home, http://processing.org/, has single-handedly put Processing on the map; the many awards and accolades bestowed upon its creators, Casey Reas and Ben Fry, haven't hurt either. One of the main reasons people continue to go to the Processing site is to visit the Processing Exhibition space (http://processing.org/exhibition/index.html), which has a simple "Add a link" feature, allowing Processors to add a link to their own Processing work. The fact that Processing sketches can be exported as Java applets is the reason this online gallery is possible.

Because Processing has been a web-based initiative, its documentation was also written in HTML and designed to take advantage of the browser environment. The Java API (application programming interface) is also HTML-based. HTML allows both Processing and Java's documentation to have embedded hyperlinks throughout, providing easy linking between related structures and concepts. The Processing API is the main language documentation for the Processing language, and can be found online at http://processing.org/reference/index.html. The Java API most useful with regard to Processing (there are a couple different ones) can be found at http://java.sun.com/j2se/1.4.2/docs/api/index.html.

Aside from the Processing API, there are two other helpful areas on the Processing site worth noting: Learning/Examples (http://processing.org/learning/index.html) and Discourse (http://processing.org/discourse/yabb_beta/YaBB.cgi). The Learning/Examples section includes numerous examples of simple Processing sketches, covering a wide variety of graphics programming topics. This section, like most of the Processing site, is an evolving archive and a great place to study well-written snippets of code as you begin learning. The Discourse section of the site includes message boards on a wide range of subjects, covering all things Processing. You'll even get replies from Casey and Ben, as well as other master Processing coders—a number of whom are well-known code artists and Processing teachers.

Hopefully by now you've successfully downloaded the Processing software. Now, let's install it and fire it up.

Launching the application

- **OS X**: After downloading the software, launch the Stuffit X archive (`.sitx`), which will create a `Processing 0124` folder. Within the folder you'll see the Processing program icon.
- **Windows**: After downloading the software, extract the ZIP archive (`.zip`), which will create a `Processing 0124` folder. Within the folder you'll see the Processing program icon.

To test that Processing is working, double-click the Processing program icon to launch the application. A window similar to the one shown in Figure 1 should open.

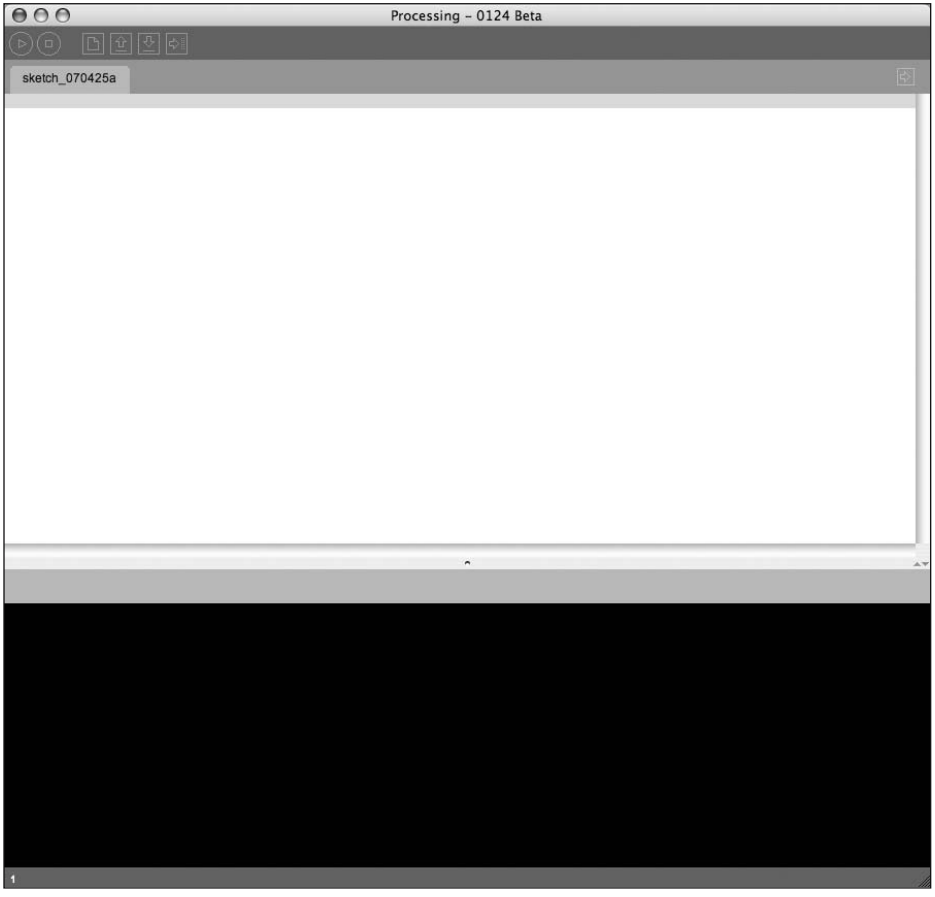

Figure 1. The Processing application interface

Processing comes with a bunch of cool code examples. Next, let's load the BrownianMotion example into the Processing application. You can access the example, and many others, through Processing's File menu, as follows:

Select File ➤ Sketchbook ➤ Examples ➤ Motion ➤ BrownianMotion from the top menu bar.

You should see a bunch of code fill the text-editor section of the Processing window, as shown in Figure 2.

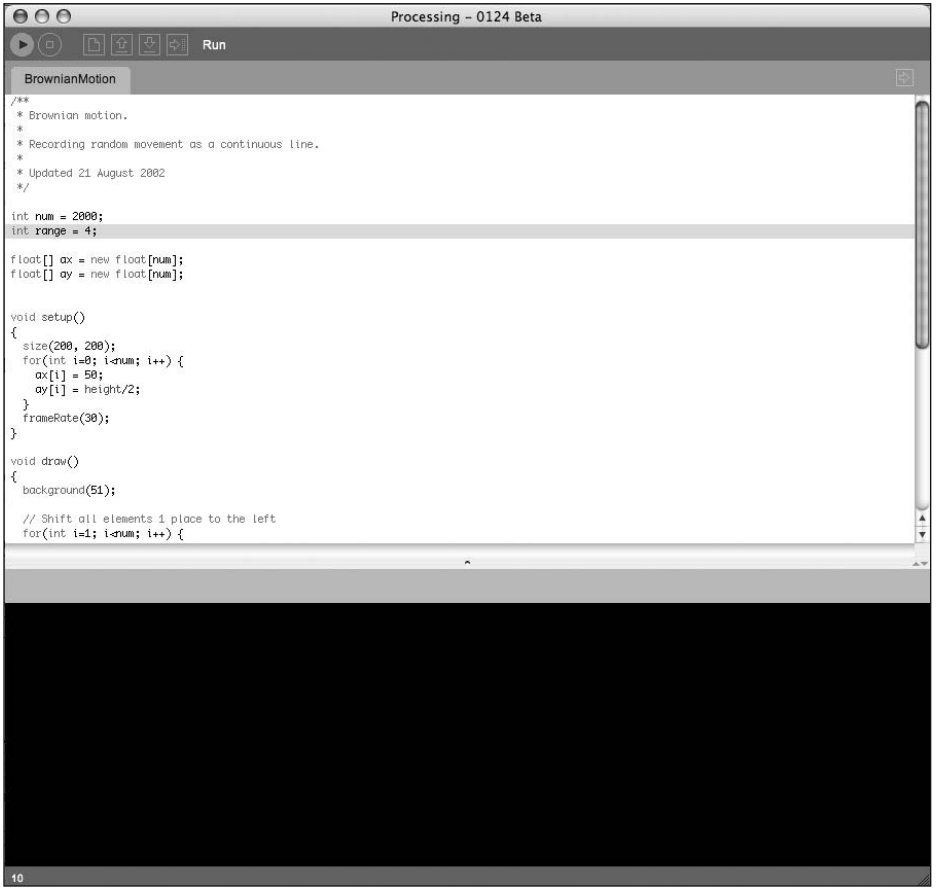

Figure 2. The Processing application interface with a loaded sketch

To launch the sketch, click the right-facing run arrow (on the left of the brown toolbar at the top of the Processing window—it looks like a VCR play button), or press Cmd+R (OS X) or Ctrl+R (Windows).

If you were successful, a 200-pixel-by-200-pixel display window with a dark gray background should have popped open, showing a white scribbly line meandering around the window (see Figure 3). Congratulations! You've just run your first Processing sketch. I recommend trying some of the other examples to get a taste of what Processing can do and to familiarize yourself a little with Processing's simple yet elegant interface.

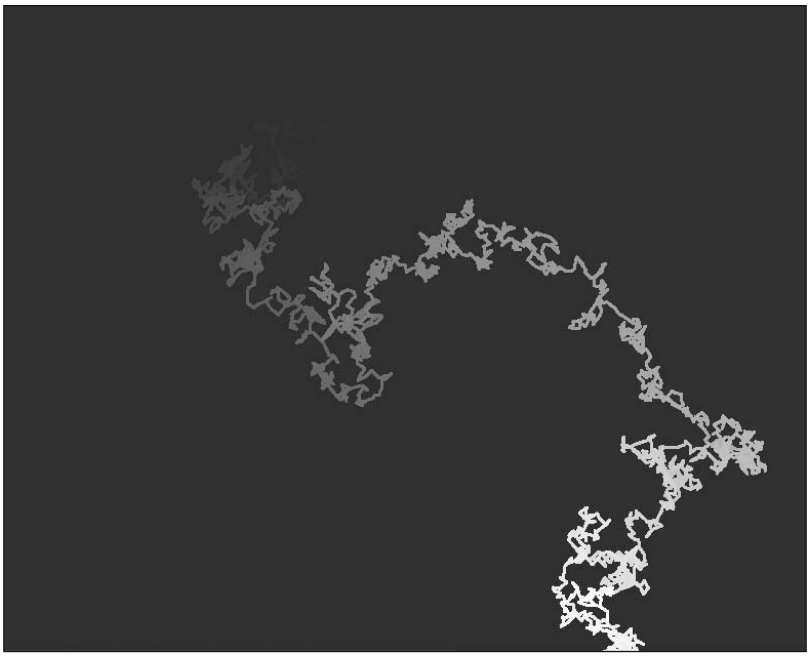

Figure 3. Screenshot of BrownianMotion sketch

How to use this book

I created this book with a couple of objectives in mind. Based on my own creative experiences working with code and application software, I wanted to present a conceptual introduction to code as a primary creative medium, including some history and theory on the subject. Based on my experiences in the digital art classroom, I wanted to provide an artist-friendly, introductory text on general programming theory and basic graphics programming. Lastly, based on my experience of working with a number of programming languages (especially ActionScript and Java), I wanted to introduce readers to an exciting new approach to creative coding with the Processing language and environment. Accomplishing all this required a fairly ambitious table of contents, which this book has.

In addition to the 800+ pages within the book, there are an additional 142 pages of "bonus" material online, at `www.friendsofed.com/book.html?isbn=159059617X`.

The bonus material is divided into Chapter 14 and Appendix C. Chapter 14 covers Processing's Java mode, as well as some advanced 3D topics. Appendix C provides a tutorial on how to use the Processing core library in "pure" Java projects—outside of the Processing environment.

In navigating all this material, I offer some suggestions to readers:

- Don't feel that you have to approach the book linearly, progressing consecutively through each chapter, or even finishing individual chapters before moving ahead. I don't think people naturally operate this way, especially not creative people. I tend to read about 20 books at a time, moving through them in some crazy fractal pattern. Perhaps my approach is too extreme, but beginning on page one of a book like this and progressing until the last page seems even more extreme. I suggest taking a grazing approach, searching for that choice patch of info to sink your brain into.
- Read stuff over and over until it sticks. I do this all the time. I often get multiple books on the same subject and read the same material presented in different ways to help me understand the material. I don't do this to memorize, but to grasp the concept.
- Don't worry about memorizing stuff you can look up. Eventually the stuff that you use a lot will get lodged in your brain naturally.
- Try to break/twist/improve my code examples. Then e-mail me your improved examples—maybe I'll use one in another book; of course I'd give you credit.
- Always keep a copy of the book in the bathroom—it's the best place to read guilt-free when the sun's still out.

Give us some feedback!

We'd love to hear from you, even if it's just to request future books, ask about friends of ED, or tell us how much you loved *Processing: Creative Coding and Computational Art*.

If you have questions about issues not directly related to the book, the best place for these inquiries is the friends of ED support forums, at `http://friendsofed.infopop.net/2/OpenTopic`. Here you'll find a wide variety of fellow readers, plus dedicated forum moderators from friends of ED.

Please direct all questions about this book to `support@friendsofed.com`, and include the last four digits of this book's ISBN (617x) in the subject of your e-mail. If the dedicated support team can't solve your problem, your question will be forwarded to the book's editors and author. You can also e-mail Ira Greenberg directly at `processing@iragreenberg.com`.

Layout conventions

To keep this book as clear and easy to follow as possible, the following text conventions are used throughout:

- Important words or concepts are normally highlighted on their first appearance in **bold type**.
- Code is presented in `fixed-width font`.

- New or changed code is normally presented in **`bold fixed-width font`**.
- Pseudocode and variable input are written in *italic fixed-width font*.
- Menu commands are written in the form Menu ➤ Submenu ➤ Submenu.
- When I want to draw your attention to something, I highlight it like this:

> *Ahem, don't say I didn't warn you.*

- Sometimes code won't fit on a single line in a book. Where this happens, I use an arrow like this: ➥

```
This is a very, very long section of code that should be written ➥
all on the same line without a break.
```