DESIGN THINKING SERIES

Coding Art

The Four Steps to Creative Programming with the Processing Language

Yu Zhang
Mathias Funk

Apress®

Design Thinking

Coding Art

The Four Steps to Creative Programming with the Processing Language

Yu Zhang
Mathias Funk

Apress®

Coding Art: The Four Steps to Creative Programming with the Processing Language

Yu Zhang
Eindhoven, The Netherlands

Mathias Funk
Eindhoven, The Netherlands

ISBN-13 (pbk): 978-1-4842-6263-4
ISBN-13 (electronic): 978-1-4842-6264-1
https://doi.org/10.1007/978-1-4842-6264-1

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Natalie Pao
Development Editor: James Markham
Coordinating Editor: Jessica Vakili

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 NY Plaza, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-6263-4. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

Table of Contents

About the Authors

Yu ZhangAn artist by training, **Yu Zhang** finished her PhD in 2017 on the theory and artistic practice of interactive technologies for public, large-scale installations. She approaches visual art with mixed reality installations and projections, sensor-based interactives, and computational arts. She roots her artistic intent in the symbolism of Asian traditions and transforms the artistic unpacking of drama and cultural signifiers into experiences of interactivity and connectivity that ultimately bridge artistic expression and audience experience. She uses systems design toolkit, to realize a complex multifaceted experience playing with the spatiotemporal context of the audience's interaction with the installations when digital and physical converge. Starting from interactivity, she constructs layers of different connections between artist, artwork, audience, and the environment to express how far such connectivity can impact and reshape the structure and relations of objects, space, and time within a dynamic audience experience. Apart from her artistic research and practice, Yu's teaching experiences cover over ten years and a broad space including traditional classrooms and design-led project-based learning activities.

Mathias Funk is Associate Professor in the Future Everyday group in the Department of Industrial Design at the Eindhoven University of Technology (TU/e). He has a background in Computer Science and a PhD in Electrical Engineering (from Eindhoven University of Technology). His research interests include complex systems design, remote data collection, systems for musical expression, and design tools such as domain-specific languages and integrated development environments. In the past, he has worked in research positions at ATR Japan, RWTH

Aachen, and he has been Visiting Researcher at Philips Consumer Lifestyle, the Netherlands. He is also the co-founder of UXsuite, a high-tech spin-off from Eindhoven University of Technology. He has years of experience in software architecture and design, engineering of distributed systems, and web technologies. Further areas of interest and practice are domain-specific languages and code generation, sound and video processing systems, and data and information visualization approaches. He has been involved extensively in the business side of innovation, the transfer of research to commercial products, and he loves to think about a design's real-world impact. As a teacher, he teaches various technology-oriented courses in the Industrial Design curriculum about designing with data and visualization approaches, systems design, and technologies for connected products and systems. He is regularly invited to give international workshops on large-scale interactive systems, group music improvisation interfaces, and expressive (musical) interaction. He has been an active musician for years and is very interested in the intersection of music, art, and design in particular.

About the Technical Reviewer

Bin Yu received his MS in biomedical engineering from Northeastern University, Shenyang, China, in 2012, and his PhD in industrial design from the Eindhoven University of Technology, in 2018. He is currently a Data Designer at Philips Design, the Netherlands, and specializes in both human–computer interaction and data visualization.

Acknowledgments

We started this book in October 2018 and went through the process of writing for several months, ending with an intensive summer writing retreat at Tenjinyama Art Studio in Sapporo. We are grateful for the hospitality and kindness of Mami Odai and her team, and we will always remember these weeks on the hill with the wind rushing through the dark trees.

From October 2019, we sent out the manuscript to reviewers, and we would like to acknowledge their hard work and sincerely thank them for great feedback and suggestions, warm-hearted encouragement, and praise: Loe Feijs (Eindhoven University of Technology), Jia Han (Sony Shanghai Creative Center), Garyfalia Pitsaki (3quarters.design), Bart Hengeveld (Eindhoven University of Technology), Joep Elderman (BMD Studio), Ansgar Silies (independent artist), and Rung-Huei Liang (National Taiwan University of Science and Technology). Without you, the book would not have been as clear and rich. We also thank the great team at Apress, Natalie and Jessica, and especially Bin Yu for his excellent technical review. Finally, we deeply appreciate the support from friends and family for this project.

CHAPTER 1

Introduction

The art world is interwoven with technology and actually quite innovative and playful. From cave paintings to the use of perspective, novel colors, and lighting, to printing techniques and direct inclusion of machines and code, there are examples of how art broke ground and changed its shape forever. Already before the beginning of the twenty-first century, artists used code and programmed machines to generate art or even be part of it.

There are so many examples of technology in art. It is also interesting to see the path of how it has grown in the past 70 years. Famous examples are, for instance, of earlier pioneers in Computer Art like Georg Nees, Michael Noll, Vera Molnár, and Frieder Nake who brought the use of pseudo-randomness and algorithm about fractals and recursion in code drawing. The young generation of artists like Casey Reas, who is well-known for developing the Processing software, extend artistic ideas through the programming language. Some artists like Jared Tarbell introduce real data into art creation and connect the complexity with the data availability. It is remarkable that for most of their works, computer artists open the source code to the public, so we can learn from them.

In this book, we want to make the point that the use of modern technology and machines in creative work does not contradict "creative expression." Instead, if used well, technology can help creatives take steps in new directions, think of new ideas, and ultimately discover their ideal form of expression.

Why data and information in art? The use of data can connect artworks to the human body, signals from outer space, or contemporary societal issues, important events happening all over the world. With data streams, creative works can become “alive.” As they represent data in visual or auditory forms, they comment on what is happening in the world; they provide an alternative frame to news and noteworthy. They can react and even create their own data as a response.

Why is interaction interesting for creatives? Interaction in an artwork opens a channel for communication with individual viewers or an entire audience. Interaction can make a work more immersive and let viewers engage in new ways with the artist’s ideas. Some might want to engage with art emotionally; some others prefer a more rational approach. The creative is in charge of defining and also limiting interactivity – from fully open access to careful limitations that preserve the overall aesthetics and message of the work. Interaction can help create multifaceted artworks that show different views on the world, or even allow for exploration of unknown territory.

Using computation and code can help a creative express ideas independent of medium and channel – the work is foremost conceptual and can be rendered in any form susceptible to the viewer. So, when we express an artistic concept in the form of code or machine instructions, we can direct the machine to produce its output in a number of ways: print a rendered image on a postcard or t-shirt, project an animation onto a building, or make an expressive interaction accessible from a single screen or for a global audience on the Internet. By disconnecting from physical matter, we create ephemeral art that might even change hands and be changed by others.

Ultimately, technology transforms what it is applied to. We show you how to do this with creativity.

1.1 Coding art

What is “coding art” all about? The title is intentionally ambiguous, ranging in meaning from how to code art to coding as creative expression. Probably the message that resonates most with you is somewhere in the middle.

Tips We are curious what you think during or after reading and working with this book. Please let us know on our website.[1]

In this book, “coding” simply means an action that translates meaning from one language into another, for example, from natural language into a computer language. This translation, as any translation, implies a change in who can and will interpret what we express in the new language. It also implies thinking about how this interpretation might work out toward a result. For natural languages, we empathize with other people, how they think and act. For machines, we need something called “computational thinking” [3, 6, 21].

Learning how to code is quite similar to learning how to speak another language. Some people might follow a more theoretical approach and learn vocabulary and grammar before attempting to speak and converse. Some others start with a conversation and gradually understand the structure of the language behind it. Depending on the circumstances, any approach might work well.

For teaching how to code in a computer or programming language, both approaches have been used in the past. There are very theoretical ways to approach coding. They often come with a steep learning curve and the full richness of what the language creators intend you to know about it. And there are also ways to playfully get used to simple examples that teach

[1] `https://codingart-book.com/feedback`

the basics before moving to more complicated examples. In the context of creative work, we strongly feel that the second approach, starting with the "conversation," works far better. However, we have seen in practice that the playful approach often hits a limitation: how to make the step from toy examples to something that is useful and also complex and intricate. This is hard and the reason why we write this book.

1.2 Motivation

Every profession, every vocation, is about doing something difficult with high quality, often using specific approaches or techniques. This works for engineers, researchers, marketing, and doing business. For creatives, the "difficult thing" is the invention of meaning and purpose out of a large set of options, constraints, and relations. It is a very human thing to create, which means we apply both our intuition and our training and knowledge to a challenge. Creatives apply various technologies in a creative process, and coding is a part of that. In this book, the use of coding in creative work is based on the situation that we try to construct meaning through understanding the logic and structure of coding. We use coding as a creative tool rather than being hardcore programmers or mere end users.

1.2.1 How to talk with a "machine"

Confronted with the particular but different characteristics of art, design, and technology, we have seen creatives struggle with questions about "how to start," "how to continue," and "how to end" while working with code and coding practice. Like writing a book or essay, it is difficult to code an idea in an individual context and condition, so that a machine can produce something meaningful for us. Unlike writing, the machine will respond swiftly to anything we feed it. It will never complain about too much work and always accurately reflect what we write in coded language.

And when we get things wrong, make a mistake, which happens more often than we are comfortable with, then this is on us. The machine is a "stupid" thing, dull and rational. Whatever creativity emerges is ours only. This book is essentially about how to let the machine express and amplify our human creativity by using precise instructions ("code") and input ("data").

For many creatives, the use of code in their projects brings new challenges, beyond successfully completing a project. For example, an unforeseen challenge is to let the work operate reliably for hours, days, and weeks. With traditional "static" material, creative output eventually turns into a stable form that rests in itself. Paper, photo, clay, concrete, metal, video, or audio documentary are stable. There are established ways to keep them safe and maintain their quality. If you want, you can study this conservation craft as a university subject even.

Things are different for art or design based on code. Code always needs a machine to run on, an environment to perform its function. This essentially counteracts technological progress: there is always a newer machine, a more modern operating system, a more powerful way to program something. Any of these get in and code written for earlier machines may stop working. This does not happen that easily to a painting or a designed and manufactured object.

1.2.2 Practice a practice

When we write about "coding" as a practice, we try to combine the creative process with computational thinking. Over the years, our art or design students, inevitably, encounter similar problems. They often ask questions like "why do we need to learn coding?", "coding is so difficult to continue once you are stuck, what is it worth?", and "I could understand the examples (from the programming software references) well, but I cannot do my idea just by using those examples, how to do that?". These questions (or often passionate complaints) point at the difficulty of learning coding as a new language. It seems that there is a big disconnect of "brainy" coding from creative practice. There is a common understanding that creative expression is fueled by inspiration and directed by intuition. In contrast, "coding" or working with technology seems to be very rational and thought through. Yes, nice try. Creative coding is only slightly "brainy" at the beginning. Soon after, it will turn into something intuitively creative and much faster than learning to wield a brush and master the skills to paint.

1.2.3 Do it and own it

Before we can start, here is yet another big "why" question: even if coding is an indispensable part of a creative project, why do artists or designers need to do the code themselves? Cooperative skills are basic for any contemporary artist and designer. Although there are cases of successful international artists who command a multidisciplinary team to work on their ideas, these people are absolutely not the norm. More realistically, we see creatives who cannot afford a team of qualified experts and who work on smaller budgets and projects. Our point in this book is: without understanding coding and technology to some extent, it will be very difficult to work with experts productively or get help when you run into problems. The point about creative technology is: you want something? Then do it and own it.

We are aware that creatives who are learning or exploring interactive art, digital art, and new media art are no longer just following one traditional approach. Instead, they need to work with their ideas from a broader perspective – in the principles of science, technology, engineering, and mathematics (STEM). When we move into the field where art meets code, creatives may need a new way of thinking and working which can help them see this new field through the lens of an old field where they have been active in and professional at.

In projects where code is involved, you as the creative need the ability to read code, understand code, perhaps even write code, and think in a computational structure. This is necessary for effectively communicating with technology experts in a common "language." We think these are essential abilities creatives today need to have. Besides, creatives who rely mostly on the help of experts often feel uncertain as to how much control they have to relinquish to achieve the goal. We actually have a section on working with technology experts towards the end of the book.

1.3 How to read this book

This book can be read in different ways, from different perspectives and also with different pre-knowledge and backgrounds. It is hard to find a common ground, but we hope that with patience and openness, you will soon see our point.

1.3.1 Calling all creatives

First of all, this book is dedicated to creatives who might be designers, artists, design or art students. We also wrote this book for architects, engineers, and researchers. They all share that creativity makes their profession special and their work unique. The creative will benefit mostly by taking the main road from beginning to end, visiting all examples and typing along. Why not bring this book to your favorite café once a week and slowly make your way through the different chapters. If you space it out over several weeks, you will see that the breaks will spark new thoughts of how to code art and what you could do yourself with the current week's topic.

We also wrote this book for educators who could take a jump to the last part first. There we explain more about the rationale behind the concepts we introduce and our methodology. We show how everything fits together, also from an educational point of view.

Third, this book is written for technical experts, who know it all actually and who might be surprised by the simplicity of the code examples. Why would they read this book? Because they realize that knowing code as a second native language and being able to construct the architecture of code is not enough, by far. The embedding of code in a process, driven by creativity or business interests, is where the challenges lie. As a technical expert, you will find the third part most interesting and can use it as a lens to scan the first two parts.

1.3.2 Four steps, one example, one zoom

In the first main part of this book, we will go through a creative process in four steps and explain how coding works in each step. The steps will each unfold through several practical examples and conclude with a short summary.

The first step, idea to visuals, gives you a short primer into working with Processing and the different visual elements that are readily available to you. We quickly proceed to working with the visual canvas before diving into animation and interaction. From this point onward, you know how to draw moving things on a canvas that might even respond to your interactive control. The second step is about composition and structure, that is, how we let art emerge from a multitude of different elements on the canvas. We will introduce data and code structure that help you in working with many visual elements at the same time. Together, we apply this in several examples around visual structure. In the third step, we show you how to work things out in more detail and how to give depth to your creations. You will learn about randomness and noise and how to control them artistically. We show you how to create smooth animations and transitions between different elements and colors. Interactivity returns in this step, and we show you how to combine interactive input with composition and refinement. The fourth step is about production, how to bring your creation to the stage, how to produce and present it well in different media from high-resolution printing to interactive installations.

On the next page, we show an example that we created inspired by an abstract geometrical painting of Kazimir Malevich ("Suprematisme," 1915) as inspiration (Figure 1-1). We chose this work because, for us, it visually hinted at a very interesting motion of otherwise static blocks that seems to be captured in a moment just before toppling over. We started with a recreation of the visual composition of ten basic elements in similar primary colors on a cream-colored canvas (step 1). In a second step, we connected to the impression of inherent motion and work with the blocks: we shifted and redrew the same composition recursively, adding more and more layers over time (step 2). The third step involved adding three large-scale rotated copies of the composition to complete the circular perspective. We also fine-tuned the timing of adding the different elements and operations over time, so the work developed in a few minutes from the first screen and visually stabilized in the last screen. Finally, we added a gradual shift of the entire canvas that, over several minutes, zoomed out and shifted the center of the canvas from the left top to the right bottom (step 3). In the fourth step, we "produce" the images that you see: we let the animation play and live select tens of frames to be automatically rendered. From these frames, we finally select eight frames as they exhibit good composition individually and also show the motion of the entire work well (step 4).

This example shows how we borrow from the four large steps described in this book, by picking a few pieces from each step that match our concept. From a process point of view, steps 1, 3, and 4 were relatively straightforward. We took more time for the second step because we went into two different directions, one more playful and one more technical, of which the playful was the right one at the end after trying both. Only after resolving this, we could move faster again. There are chances that you will struggle as well while working with this book; don't forget to take breaks and never let go.

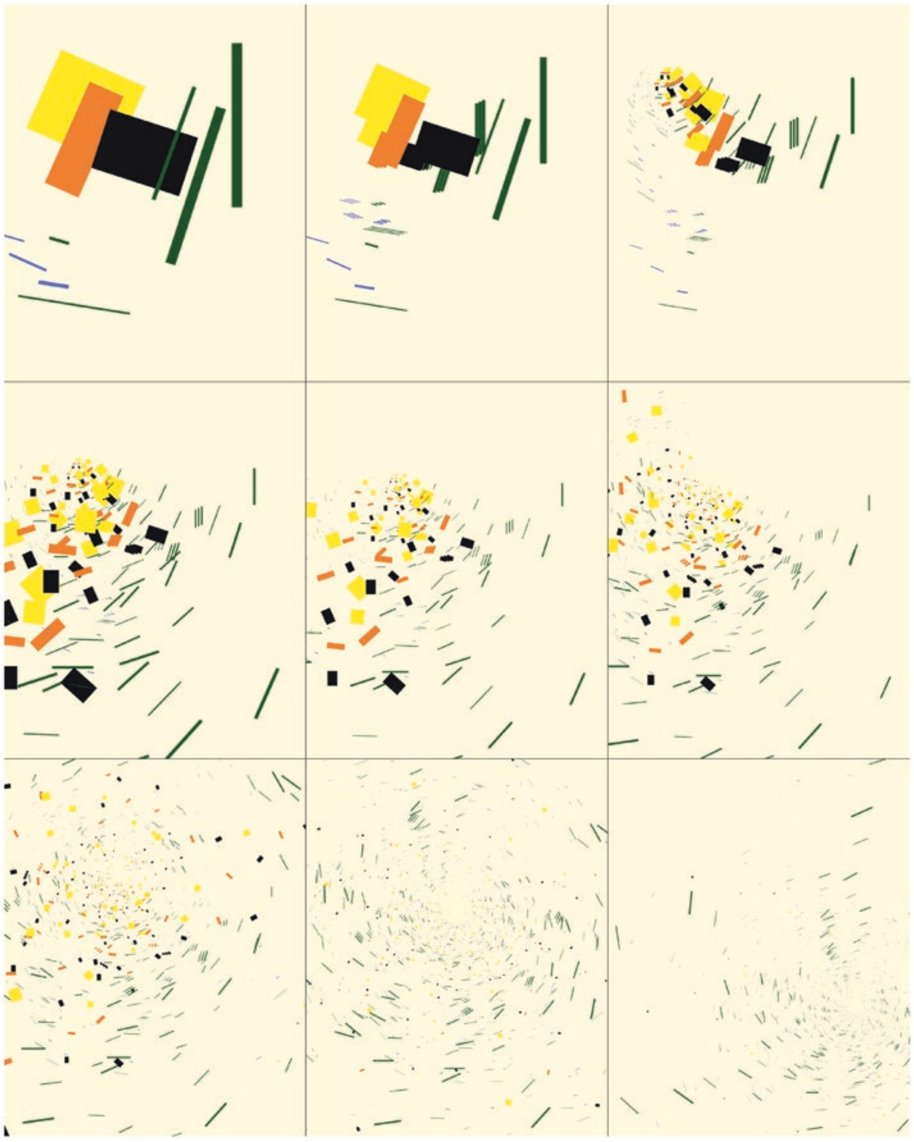

Figure 1-1. *Example of generative art taking an abstract geometrical painting of Kazimir Malevich ("Suprematisme," 1915) as inspiration*

Throughout these four steps, we will teach you about creative computation, and, at some point, you will see also bits of strategies, patterns, and more complex concepts appear. Afterward, we will roll up all steps in a larger art project, MOUNTROTHKO, in the second part of this book. Finally, in the third part, we zoom out and turn toward the practice of creative coding, through learning and collaboration. This part shows you how you can make progress using this book and beyond, what you can do when you feel stuck, and how to get help. It's all there, you just need to go step by step toward it.

1.3.3 Getting ready

This book contains a lot of examples, and they are written in code ("source code"). Most examples can be used directly, and the resulting visual output is shown close to the source code.

Code examples

```
// How to quickly find code examples in the book?

Look for text in a box like this!
```

All source code listed in this book is written in the open source software Processing. Processing itself is available from `https://processing.org`, and we recommend that you install it on your computer to get the most out of this book. Processing is a medium for understanding the structure and logic of code. We will explain this shortly. The code examples are available online from our Processing library.[2]

[2]The Processing library can be found here: `https://codingart-book.com/library`. You can install it using the Processing library manager.

Although it might be tempting to just download the examples and play with them, we recommend typing them yourself (at least some of them). This way, you will pick up the programming style much faster and allow your muscle memory to support your learning. And if you are lucky, you will make a few small mistakes that give you surprising results.

Finally, we will address you, the reader, informally. Think of this book as a conversation in your favorite café over coffee and your laptop is right in front of you. Feel free to pause the conversation and dive into a topic on your own, or explore the code of the examples, and then resume to the next page. Let’s begin.

PART I

Creative coding

CHAPTER 2

Idea to visuals

In this first part of the book, we will go through four process steps and show for each step how coding becomes a meaningful part of our creative process. In step 1, IDEA TO VISUALS, we take a bottom-up approach and start directly with visuals and code. Our entry point to this approach is to use code directly from the ideation stage of the creative process. More specifically, instead of making mood boards, sketching, writing, searching the web, or talking to experts, we suggest that you just start Processing and give it a spin. First, we look at how we can express our ideas using Processing and a few lines of code. Yes, we start really simple.

2.1 Visual elements

For many artists, even if visual elements in their work are coded, the standards for effectiveness in their work are still based on either cognitive or aesthetic goals [12, 18, 20]. When we analyze any drawings, paintings, sculptures, or designs, it is similar – we examine and decompose them to see how they are put together to create the overall effect of the work. Lines, colors, shapes, scale, form, and textures are the general fundamental components of aesthetics and cognition for both art and design and for coding art as well.

Processing can draw a wide range of different forms that result from variation and combination of simple shapes. When you take an example from the Processing reference, try to change the numbers in the example to explore how the shape changes and responds to different numbers.