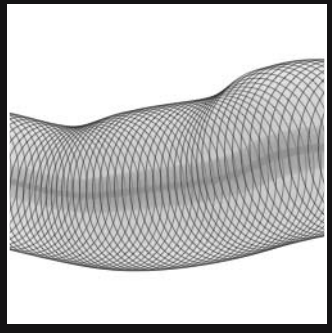
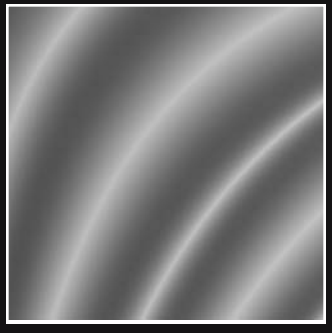
10 COLOR AND IMAGING


Thus far in the book I've focused on the structural aspects of creative coding—an armature upon which to build. However, I've barely scraped the surface of Processing's expressive potential. Nowhere is this potential more evident than in Processing's extensive color and imaging capabilities. I'll begin this chapter by exploring Processing's high-level color and imaging functions, which conceal the internal mathematics while still providing lots of creative range. Working at the pixel level, you'll learn how to generate custom gradients and even spin some of your own cool imaging filters. Finally, you'll apply an object-oriented approach to imaging and color, building upon the OOP concepts introduced in Chapters 8 and 9.

The importance of color

One of the major contributing factors in creating expressive work is color. This is not in any way to imply that powerfully expressive work can't be done in grayscale—artists such as Francisco de Goya, Franz Kline, and William Kentridge immediately come to mind. However, I've found that I have a different relationship with my work and process when I work in color. As a painter, I found I could express a wider range of thoughts and feelings when my palette included a full range of color, as opposed to just grayscale. I think this holds true for creative coding as well.

In the beginning, when you are first learning the basics of coding, the whole process might seem purely analytical, but like any discipline, once you develop some fluency, certain processes go on autopilot, freeing you to access other cognitive states. Coding grayscale normally uses 8 bits of information per pixel, providing 256 different values from black to white—black being 0 and white being 255. Color generally uses 24 bits of information per pixel (or 256 × 256 × 256, giving you 16,777,216 different colors). In addition, an extra 8 bits of data is often used for alpha, or transparency. At the risk of sounding overly new-agey, color seems to provide us with a wider range of responses to map to a wider range of emotions than grayscale does. Once we begin to explore color at the pixel level we are only a few steps away from digital imaging, or using code and procedural processing to affect all or a range of the pixels making up a continuous tone image—like a photograph. Processing comes with a bunch of cool functions for doing just this. In addition, this chapter will explore how to spin some of your own imaging filters. To begin, though, let's look at some simple color examples (see Figure 10-1):

```
// Color Shift
size(800, 450);
background(50);
noStroke();
int spacing = 50;
int w = (width-spacing*2)/2;
int h = w;
color swatch = color(100, 100, 120);
```