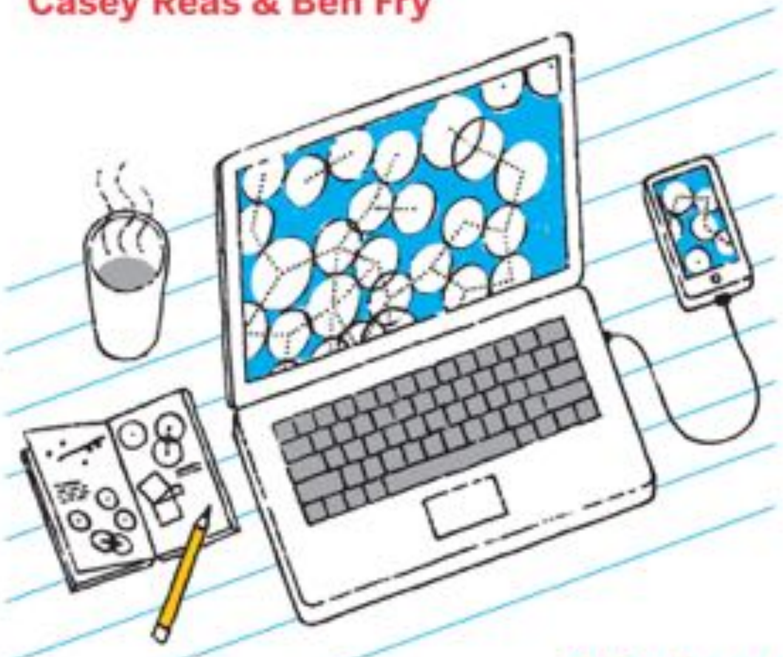
all code illustrations taken from the processing.org website +

Make: PROJECTS

A HANDS-ON INTRODUCTION TO MAKING INTERACTIVE GRAPHICS

Getting Started with Processing

Casey Reas & Ben Fry



O'REILLY

Make: makezine.com

Open Source/Graphics Programming

Make: PROJECTS

Getting Started with Processing

Learn computer programming the easy way with Processing, a simple language that lets you use code to create drawings, animation, and interactive graphics. Programming courses usually start with theory, but this book lets you jump right into creative and fun projects. It's ideal for anyone who wants to learn programming, and serves as a simple introduction to graphics for people who already have some programming skills.

Written by the founders of Processing, this book takes you through the learning process one step at a time to help you grasp core programming concepts. Join the thousands of hobbyists, students, and professionals who have discovered this free and educational community platform.

- Quickly learn programming basics, from variables to objects
- Understand the fundamentals of computer graphics
- Get acquainted with the Processing software environment
- Create interactive graphics with easy-to-follow projects
- Use the Arduino open source prototyping platform to control your Processing graphics

Casey Reas is a professor in the Department of Design Media Arts at UCLA and a graduate of the MIT Media Laboratory. Reas's software has been featured in numerous solo and group exhibitions in the U.S., Europe, and Asia.

Ben Fry, a designer, programmer, and author based in Cambridge, Massachusetts, received his doctoral degree from the MIT Media Laboratory. He worked with Casey Reas to develop Processing, which won a Golden Nica from the Prix Ars Electronica in 2005.

Make: makezine.com

O'REILLY

US $19.99 CAN $24.99
ISBN: 978-1-449-37980-3
51999
9 781449 379803

10/Arrays

We've introduced new programming ideas in each chapter (variables, functions, objects) and now we've come to the last step—arrays! An *array* is a list of variables that share a common name. Arrays are useful because they make it possible to work with more variables without creating a new name for each. This makes the code shorter, easier to read, and more convenient to update.

+

9/Objects

Object-oriented programming (OOP) is a different way to think about your programs. Although the term "object-oriented programming" may sound intimidating, there's good news: you've been working with objects since Chapter 6, when you started using *PImage*, *PFont*, *String*, and *PShape*. Unlike the primitive data types *boolean*, *int*, and *float*, which can store only one value, an object can store many. But that's only a part of the story. Objects are also a way to group variables with related functions. Because you already know how to work with variables and functions, objects simply combine what you've already learned into a more understandable package.

Make an Array

Each item in an array is called an *element*, and each has an *index* value to mark its position within the array. Just like coordinates on the screen, index values for an array start counting from 0. For instance, the first element in the array has the index value 0, the second element in the array has the index value 1, and so on. If there are 20 values in the array, the index value of the last element is 19. Figure 10-1 shows the conceptual structure of an array.

```
int[] years = { 1920, 1972, 1980, 1996, 2010 };
```

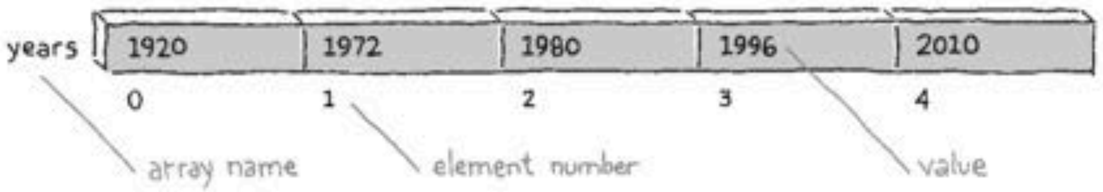


Figure 10-1. An array is a list of one or more variables that share the same name.

Example 10-4: Declare and Assign an Array

First we'll declare the array outside of *setup()* and then create and assign the values within. The syntax *x[0]* refers to the first element in the array and *x[1]* is the second:

```
int[] x;           // Declare the array

void setup() {
  size(200, 200);
  x = new int[2];  // Create the array
  x[0] = 12;       // Assign the first value
  x[1] = 2;        // Assign the second value
}
```

http://examples.oreilly.com/0636920000570/interactive_examples/index.php?example=Ex_10_04

Example 10-5: Compact Array Assignment

Here's a slightly more compact example, in which the array is both declared and created on the same line, then the values are assigned within *setup()*:

```
int[] x = new int[2];  // Declare and create the array

void setup() {
  size(200, 200);
  x[0] = 12;           // Assign the first value
  x[1] = 2;            // Assign the second value
}
```

Store the last 60 positions of the mouse, and draw them in some interesting way.

Build a bouncy ball class, and organize (store) them with an array

Add a trail to the bouncy ball class

Typical Class Structure:

```
class JitterBug {

  float x;
  float y;
  int diameter;
  float speed = 2.5;

  JitterBug(float tempX, float tempY, int tempDiameter) {
    x = tempX;
    y = tempY;
    diameter = tempDiameter;
  }

  void move() {
    x += random(-speed, speed);
    y += random(-speed, speed);
  }

  void display() {
    ellipse(x, y, diameter, diameter);
  }

}
```

Block

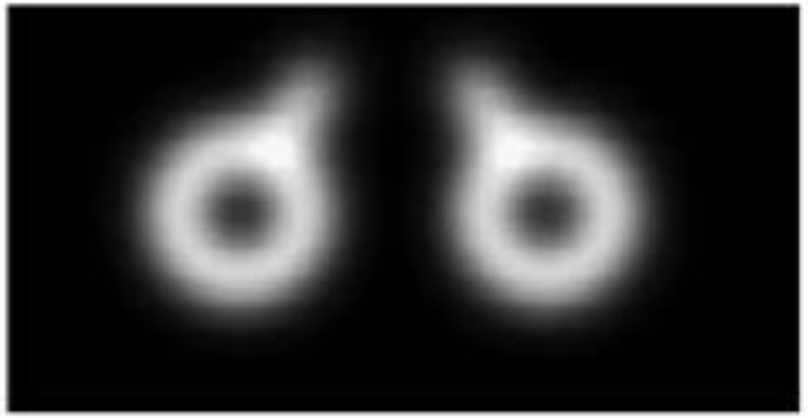
Fields

Constructor

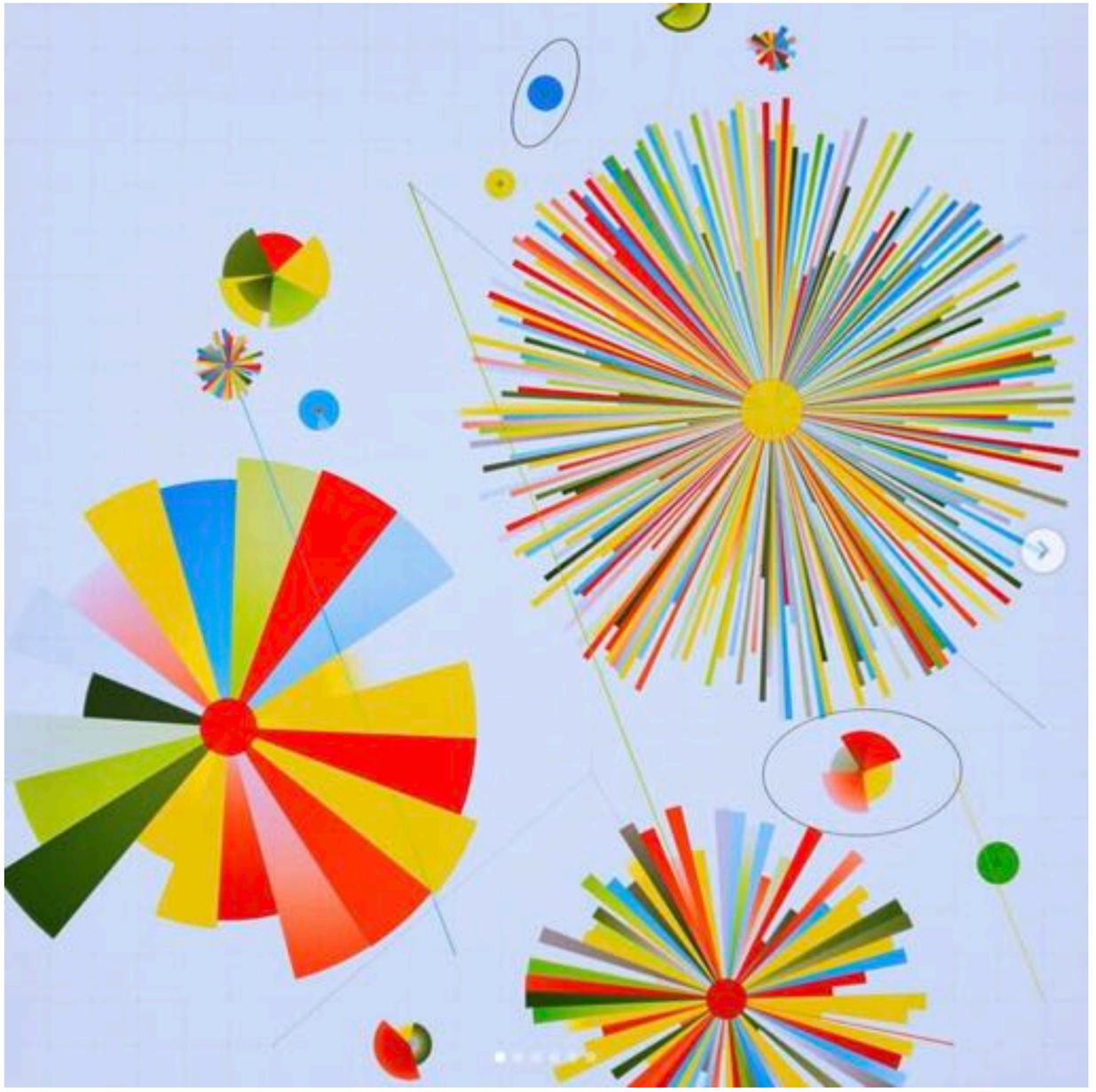
Methods

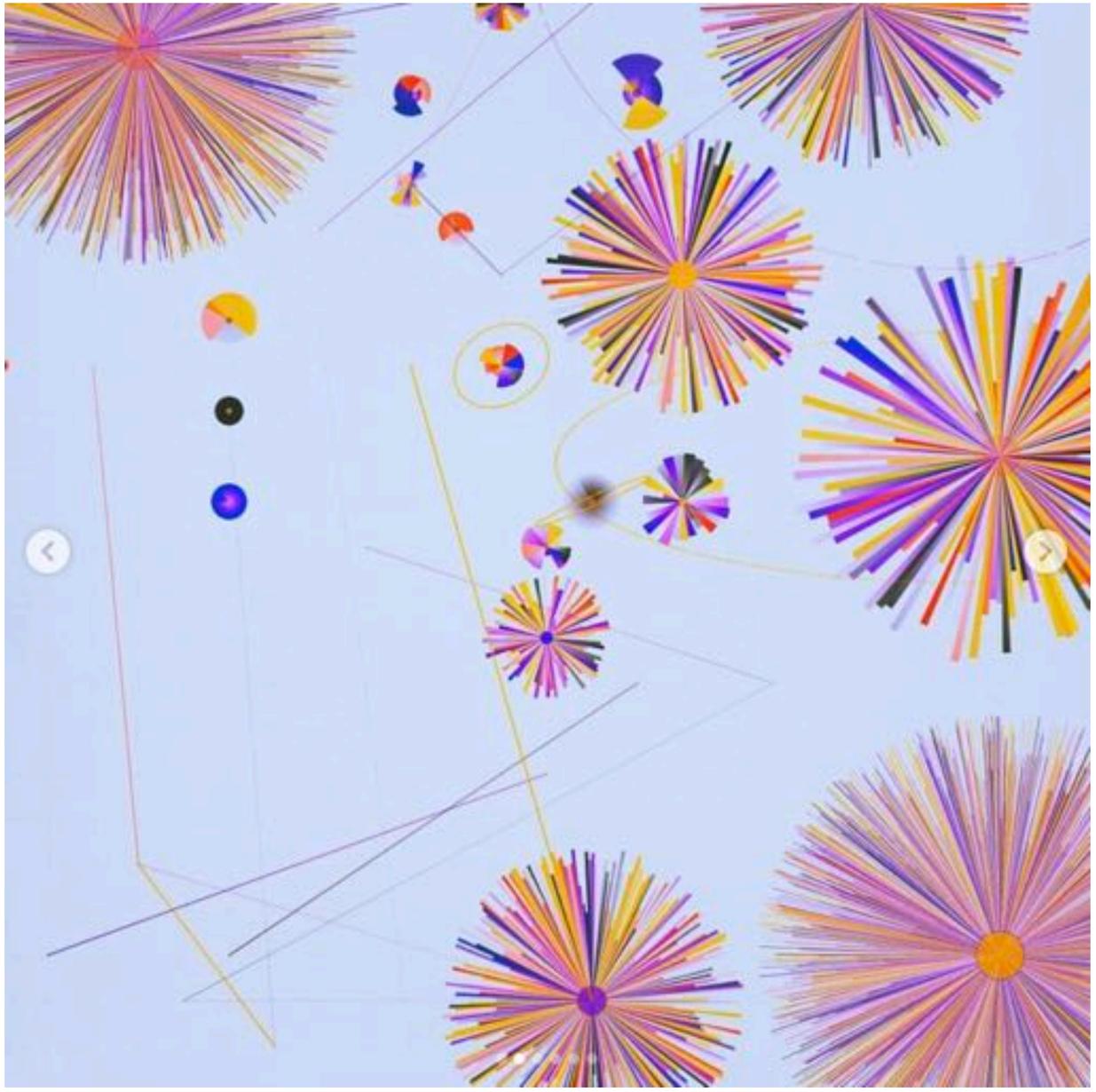
Example 10-11: Sequences of Images

To run this example, get the images from the *media.zip* file as described in Chapter 6. The images are named sequentially (*frame-0000.png*, *frame-0001.png*, and so forth), which makes it possible to create the name of each file within a *for* loop, as seen in the eighth line of the program:



A Look at Architecture **(Dancing C's) by Paul Rand (1972-1984)**
https://www.wright20.com/auctions/2018/09/paul-rand-the-art-of-design/222





Instagram Posts by Manoloide (2018)
https://www.instagram.com/Manoloide/

Homework 09 // Due 2018.11.05

Use an array to either create a trail behind the mouse or behind several objects (the array would be within the class), and develop the visuals in an interesting way.

Size: 1280x720 pixels or 720x720 pixels

Expand your work from last week (Homework 08) to include an array to store and organize your objects. Feel free to create a new class to work with if you've already used an array on the previous assignment.

Size: 1280x720 pixels or 720x720 pixels