USC IML 288 __ 2018.08.27
1/Hello, 2/Starting to Code, 3/Draw

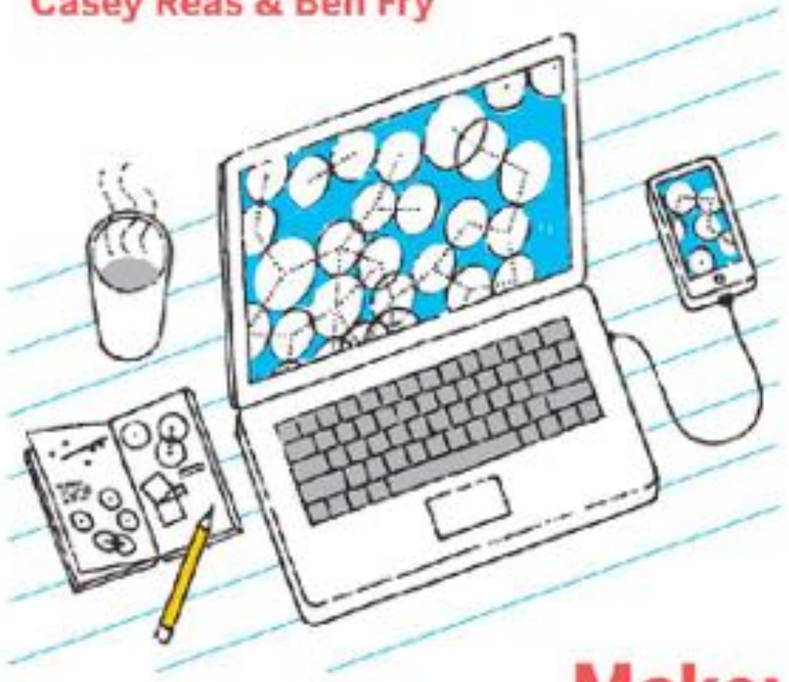
all code illustrations taken from the processing.org website +

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

Preface

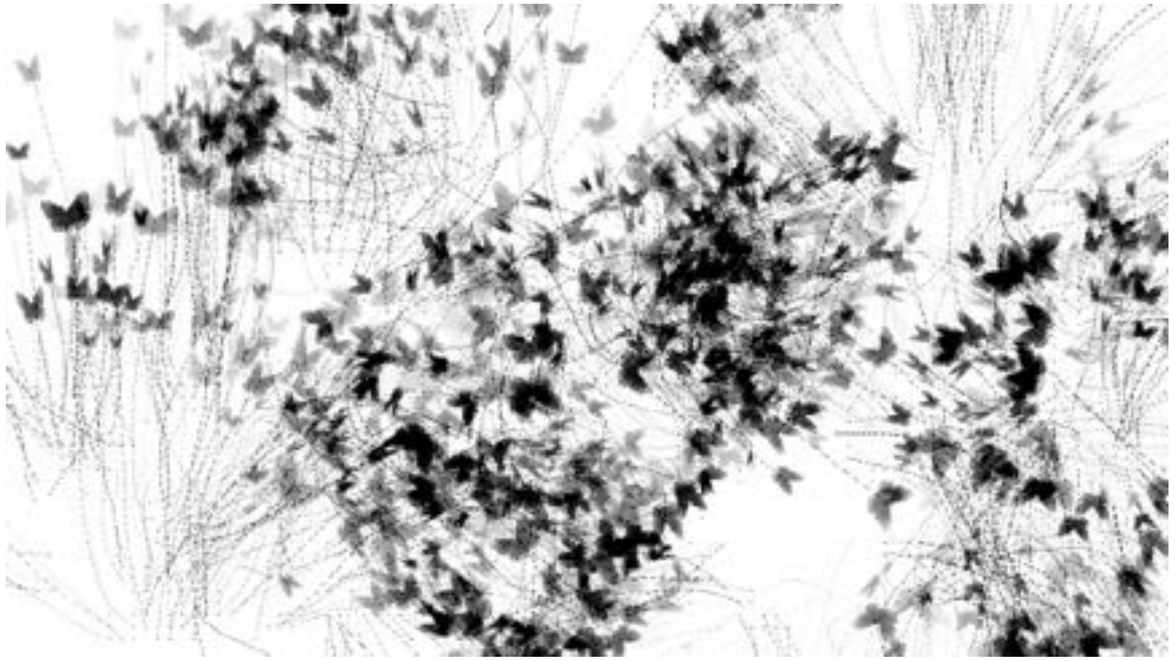
We created Processing to make programming interactive graphics easier. We were frustrated with how difficult it was to write this type of software with the programming languages we usually used (C++ and Java) and were inspired by how simple it was to write interesting programs with the languages of our childhood (Logo and BASIC). We were most influenced by Design By Numbers (DBN), a language created by our research advisor, John Maeda, which we were maintaining and teaching at the time.

1/Hello

Processing is for writing software to make images, animations, and interactions. The idea is to write a single line of code, and have a circle show up on the screen. Add a few more lines of code, and the circle follows the mouse. Another line of code, and the circle changes color when the mouse is pressed. We call this *sketching* with code. You write one line, then add another, then another, and so on. The result is a program created one piece at a time.

Sketching and Prototyping

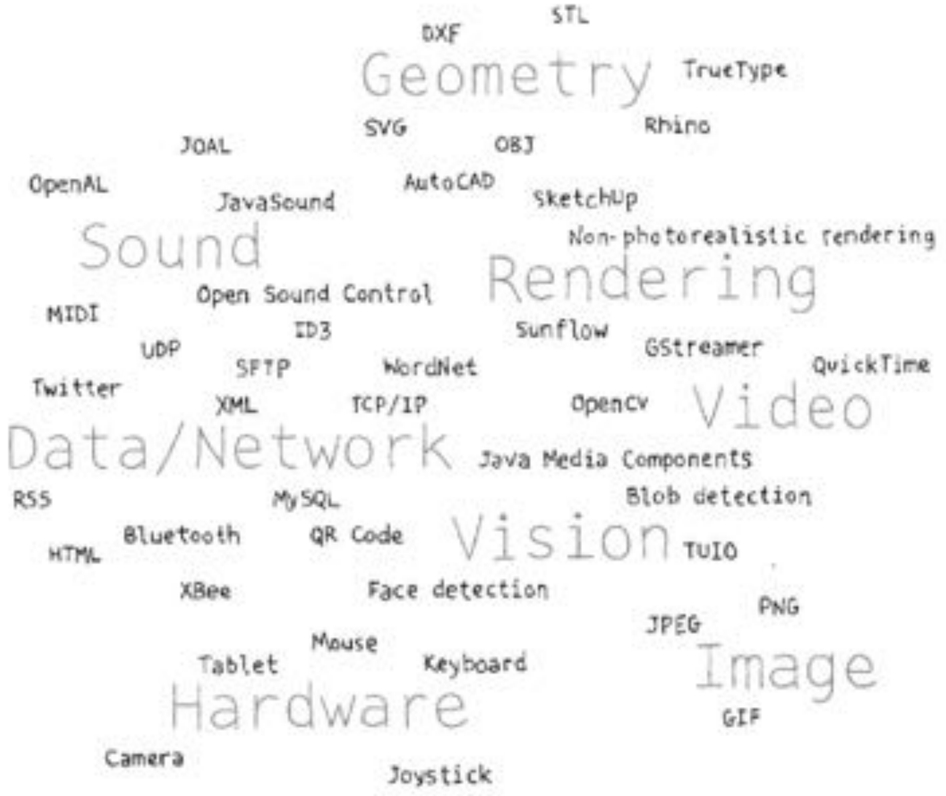
Sketching is a way of thinking: it's playful and quick. The basic goal is to explore many ideas in a short amount of time. In our own work, we usually start by sketching on paper and then moving the results into code. Ideas for animation and interactions are usually sketched as storyboards with notations. After making some software sketches, the best ideas are selected and combined into prototypes (Figure 1-1). It's a cyclical process of making, testing, and improving that moves back and forth between paper and screen.



butterflies for binky, 2018

Flexibility

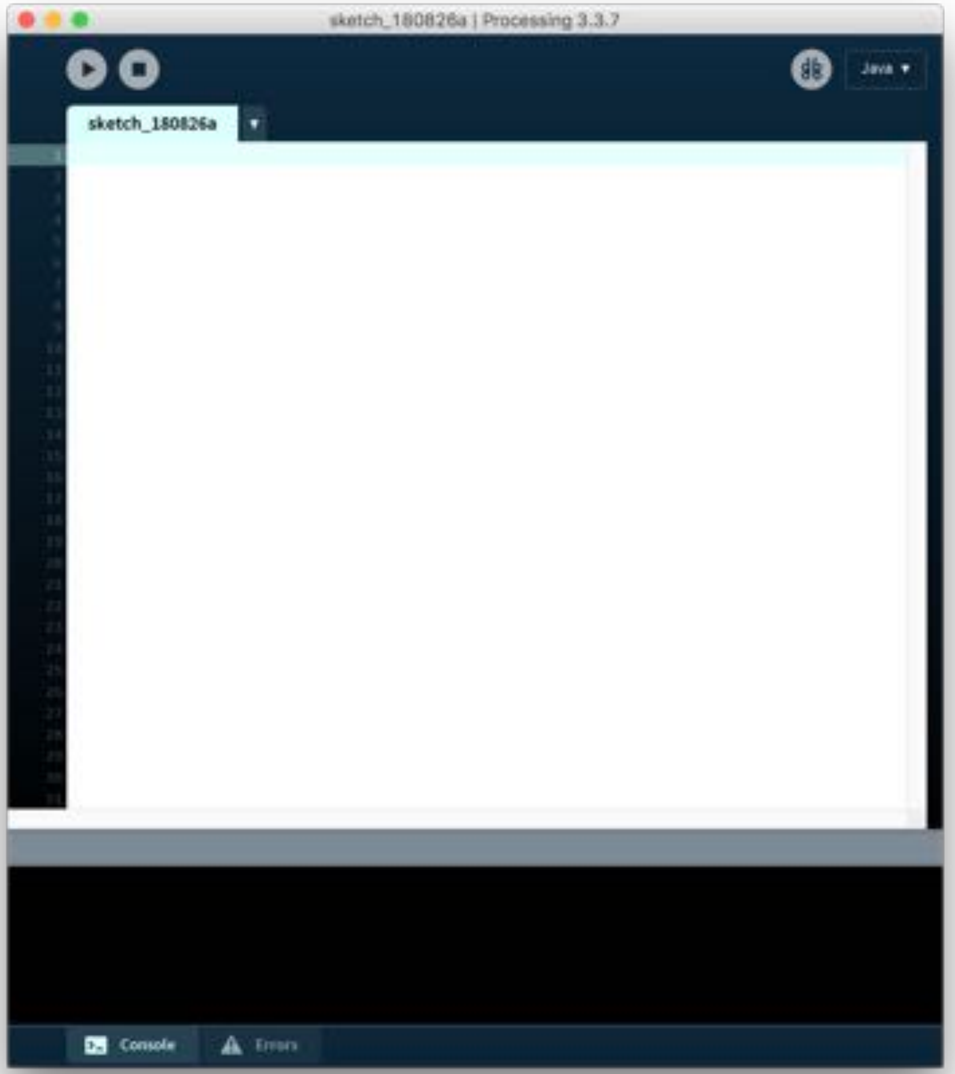
Like a software utility belt, Processing consists of many tools that work together in different combinations. As a result, it can be used for quick hacks or for in-depth research. Because a Processing program can be as short as one line or as long as thousands, there's room for growth and variation. More than 100 libraries extend Processing even further into domains including sound, computer vision, and digital fabrication (Figure 1-2).



2/Starting to Code

To get the most out of this book, you need to do more than just read the words. You need to experiment and practice. You can't learn to code just by reading about it—you need to do it. To get started, download Processing and make your first sketch.

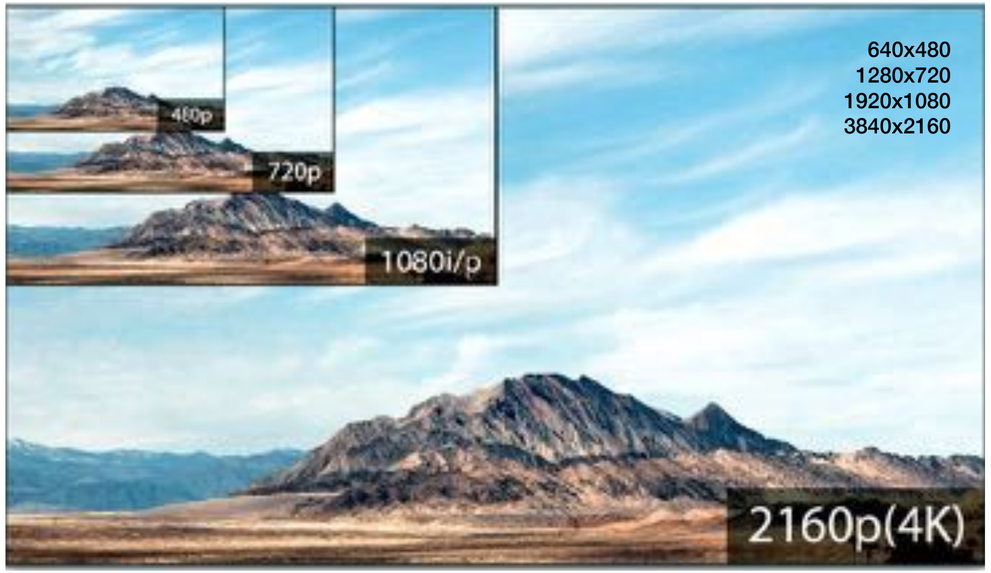
Save



3/Draw

At first, drawing on a computer screen is like working on graph paper. It starts as a careful technical procedure, but as new concepts are introduced, drawing simple shapes with software expands into animation and interaction. Before we make this jump, we need to start at the beginning.

```
size(480, 120);
point(240, 60);
```



https://www.lifewire.com/4k-resolution-overview-and-perspective-1846842

Display

Super Retina HD display

5.8-inch (diagonal) all-screen OLED Multi-Touch display

HDR display

2436-by-1125-pixel resolution at 458 ppi

1,000,000:1 contrast ratio (typical)

True Tone display

Wide color display (P3)

3D Touch

625 cd/m2 max brightness (typical)

Fingerprint-resistant oleophobic coating

Support for display of multiple languages and characters simultaneously

The iPhone X display has rounded corners that follow a beautiful curved design, and these corners are within a standard rectangle. When measured as a standard rectangular shape, the screen is 5.85 inches diagonally (actual viewable area is less).

http://www.apple.com

Display

Retina display

15.4-inch (diagonal) LED-backlit display with IPS technology; 2880-by-1800 native resolution at 220 pixels per inch with support for millions of colors

Supported scaled resolutions:

- 1920 by 1200
- 1680 by 1050
- 1280 by 800
- 1024 by 640

500 nits brightness

Wide color (P3)

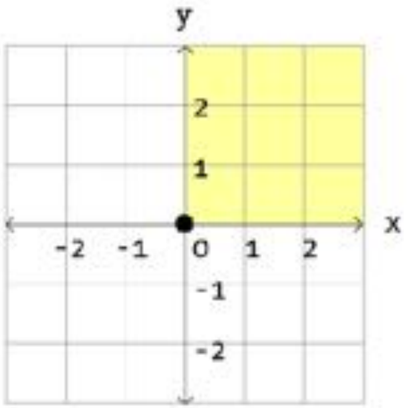
True Tone technology

https://www.processing.org/tutorials/drawing

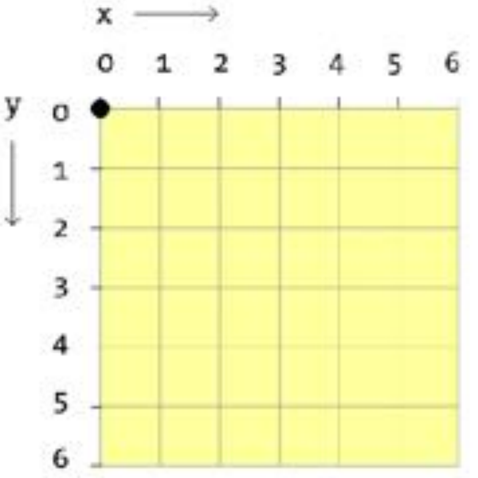
Coordinate System and Shapes

Daniel Shiffman

Before we begin programming with Processing, we must first channel our eighth grade selves, pull out a piece of graph paper, and draw a line. The shortest distance between two points is a good old fashioned line, and this is where we begin, with two points on that graph paper.



Eighth Grade



Computer

3/Draw

At first, drawing on a computer screen is like working on graph paper. It starts as a careful technical procedure, but as new concepts are introduced, drawing simple shapes with software expands into animation and interaction. Before we make this jump, we need to start at the beginning.

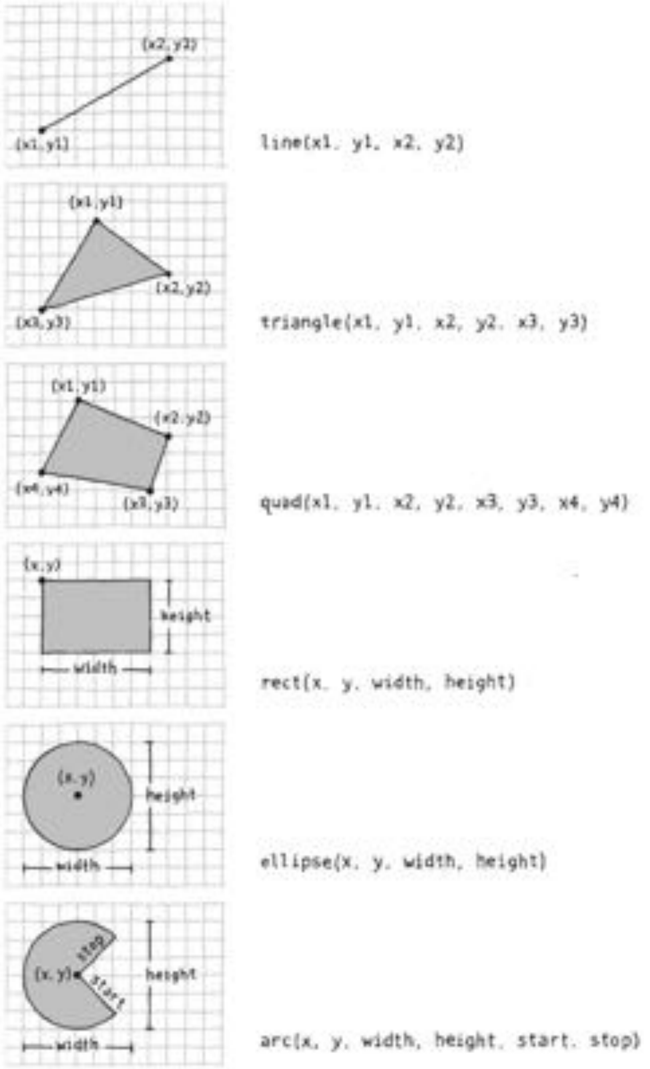


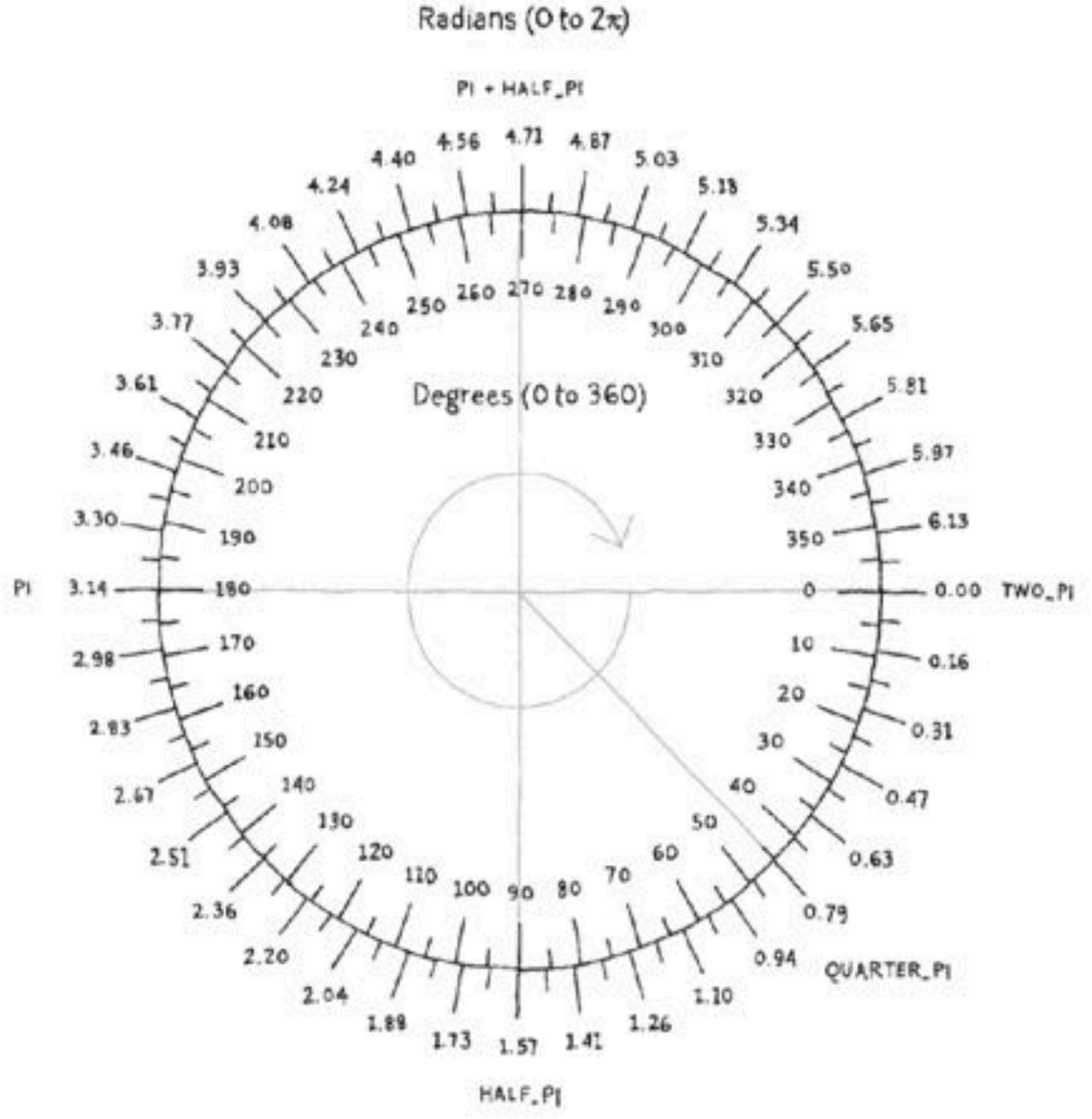
Figure 3-1. Coordinates and shapes.

```
size(480, 120);
rect(180, 60, 220, 40);
```

```
size(480, 120);
ellipse(278, -100, 400, 400);
ellipse(120, 100, 110, 110);
ellipse(412, 60, 18, 18);
```

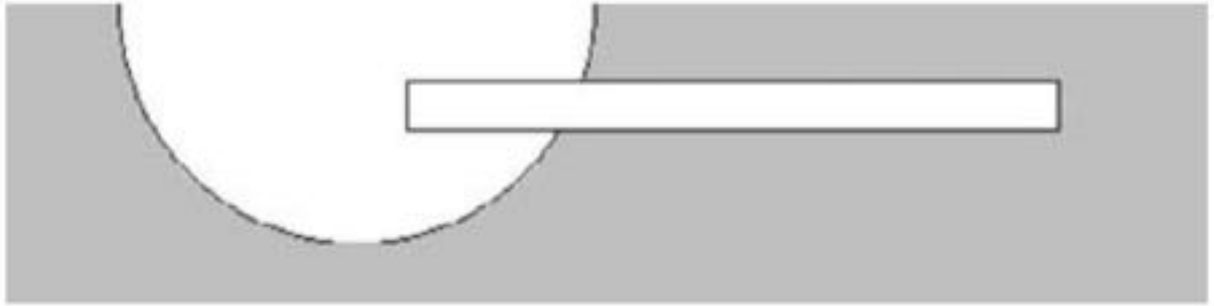
```
size(480, 120);
arc(90, 60, 80, 80, 0, HALF_PI);
arc(190, 60, 80, 80, 0, PI+HALF_PI);
arc(290, 60, 80, 80, PI, TWO_PI+HALF_PI);
arc(390, 60, 80, 80, QUARTER_PI, PI+QUARTER_PI);
```

```
size(480, 120);
arc(90, 60, 80, 80, 0, radians(90));
arc(190, 60, 80, 80, 0, radians(270));
arc(290, 60, 80, 80, radians(180), radians(450));
arc(390, 60, 80, 80, radians(45), radians(225));
```

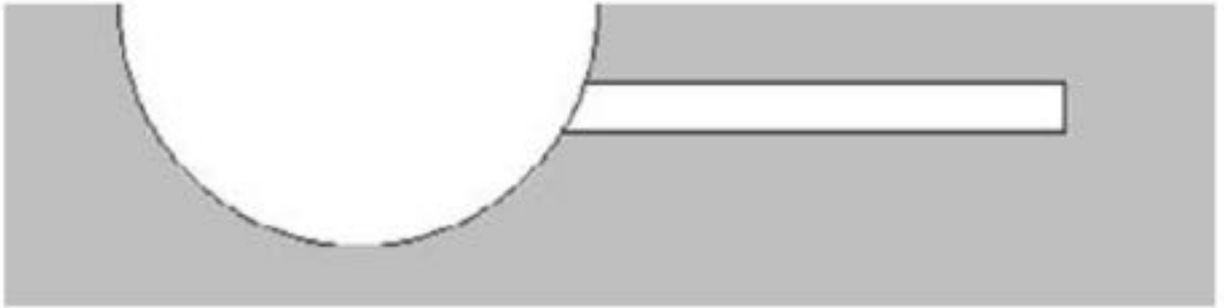


```
size(480, 120);
beginShape();
vertex(180, 82);
vertex(207, 36);
vertex(214, 63);
vertex(407, 11);
vertex(412, 30);
vertex(219, 82);
vertex(226, 109);
endShape();
```

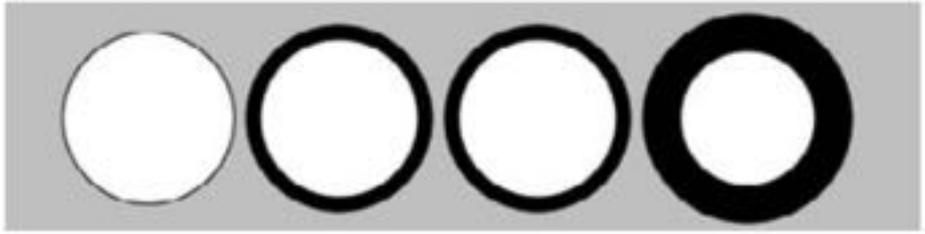
Example 3-9: Control Your Drawing Order



```
size(480, 120);
ellipse(140, 0, 190, 190);
// The rectangle draws on top of the ellipse
// because it comes after in the code
rect(160, 30, 260, 20);
```



```
size(480, 120);
rect(160, 30, 260, 20);
// The ellipse draws on top of the rectangle
// because it comes after in the code
ellipse(140, 0, 190, 190);
```



```
size(480, 120);
smooth();
ellipse(75, 60, 90, 90);
strokeWeight(8);  // Stroke weight to 8 pixels
ellipse(175, 60, 90, 90);
ellipse(279, 60, 90, 90);
strokeWeight(20);  // Stroke weight to 20 pixels
ellipse(389, 60, 90, 90);
```

```
size(480, 120);
smooth();
strokeWeight(12);
strokeJoin(ROUND);       // Round the stroke corners
rect(40, 25, 70, 70);
strokeJoin(BEVEL);       // Bevel the stroke corners
rect(140, 25, 70, 70);
strokeCap(SQUARE);       // Square the line endings
line(270, 25, 340, 95);
strokeCap(ROUND);        // Round the line endings
line(350, 25, 420, 95);
```

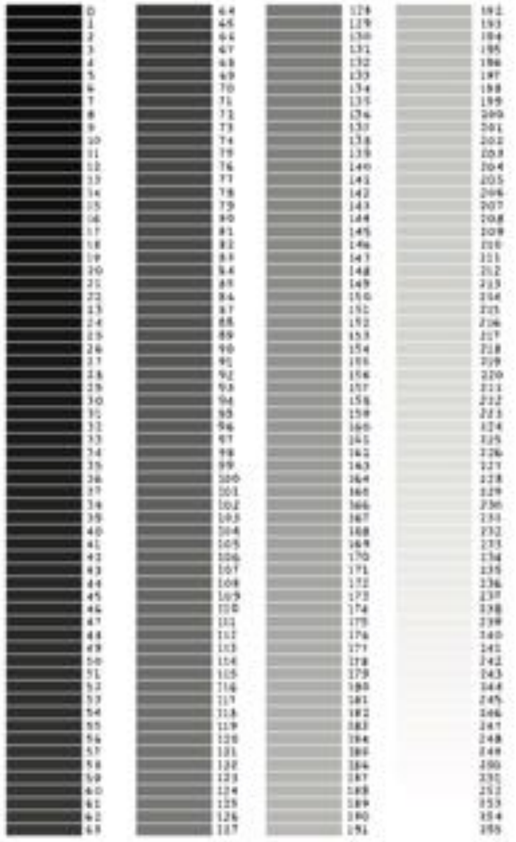


Figure 3-3. Gray values from 0 to 255.

This example shows three different gray values on a black background:

```
size(480, 120);
smooth();
background(0);                // Black
fill(204);                    // Light gray
ellipse(132, 82, 200, 200);   // Light gray circle
fill(153);                    // Medium gray
ellipse(228, -16, 200, 200); // Medium gray circle
fill(102);                    // Dark gray
ellipse(268, 118, 200, 200); // Dark gray circle
```

https://www.processing.org/tutorials/color

Processing

Color

Daniel Shiffman

In the digital world, when we want to talk about a color, precision is required. Saying "Hey, can you make that circle bluish-green?" will not do. Color, rather, is defined as a range of numbers. Let's start with the simplest case: black & white or grayscale. 0 means black, 255 means white. In between, every other number—50, 87, 162, 209, and so on—is a shade of gray ranging from black to white.

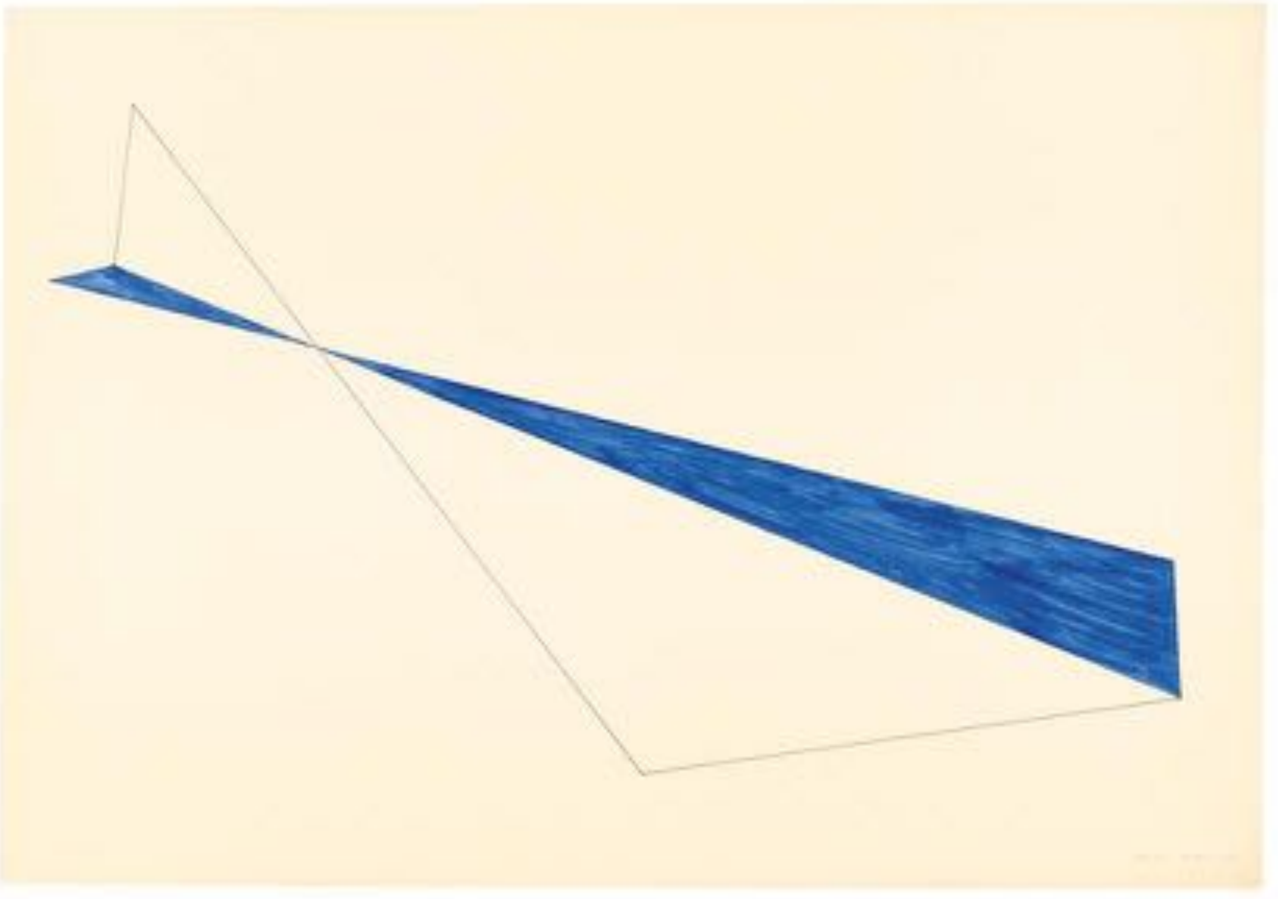
Does 0-255 seem arbitrary to you?
Color for a given shape needs to be stored in the computer's memory. This memory is just a long sequence of 0's and 1's (a whole bunch of on or off switches.) Each one of these switches is a bit, eight of them together is a byte. Imagine if we had eight bits (one byte) in sequence—how many ways can we configure these switches? The answer is (and doing a little research into binary numbers will prove this point) 256 possibilities, or a range of numbers between 0 and 255. We will use eight bit color for our grayscale range and 24 bit for full color (eight bits for each of the red, green, and blue color components).

Comments

The examples in this chapter use double slashes (//) at the end of a line to add comments to the code. Comments are parts of the program that are ignored when the program is run. They are useful for making notes for yourself that explain what's happening in the code. If others are reading your code, comments are especially important to help them understand your thought process.

JOSEF ALBERS, *HOMAGE TO THE SQUARE*, 1965

http://www.albersfoundation.org/art/josef-albers/paintings/homages-to-the-square/index/



ANNE TRUITT, *14 JUNE '65* [1], 1965

https://www.matthewmarks.com/new-york/exhibitions/2015-09-11_anne-truitt-in-japan/works-in-exhibition/

ELLSWORTH KELLY AT SFMOMA

https://www.sfmoma.org/artist/Ellsworth_Kelly

Homework 01 // Due 2018.09.10

Part I. Recreate a Form and Color / Abstract Minimalist Artwork with Code
and think about draw order, color, form, shapes, fill, stroke.

Size: appropriate to the work and fits within 1280x720 pixels
(e.g. a square work would be 720x720 pixels)

Part II. Develop a Concept for Your Own Work
Inspired by Part I

Size: appropriate to the work and fits within 1280x720 pixels